Technical Report

DTRA-TR-00-18

Digital Communications in Fading and Jamming–COMLINK Users' Manual

Approved for public release; distribution is unlimited.

April 2002

Prepared for:
Defense Threat Reduction Agency
8725 John J. Kingman Road, MS-6201
Fort Belvoir, VA 22060-6201

DTRA01-99-C-0035

Robert L. Bogusch

Prepared by:
Mission Research Corporation
P. O. Box 2256
Atascadero, CA 93423-2256

DISTRIBUTION LIST UPDATE

This mailer is provided to enable DTRA to maintain current distribution lists for reports. (We would appreciate you providing the requested information.)

- ☐ Add the individual listed to your distribution list.
- ☐ Delete the cited organization/individual.
- ☐ Change of address.

Note:
Please return the mailing label from the document so that any additions, changes, corrections or deletions can be made easily. For distribution cancellation or more information call DTRA/IMCI (703 767-4726.

NAME: ____________________

ORGANIZATION: ____________________

OLD ADDRESS	**NEW ADDRESS**
____________________	____________________
____________________	____________________
____________________	____________________

TELEPHONE NUMBER: ( ) ____________________

DTRA PUBLICATION NUMBER/TITLE	**CHANGES/DELETIONS/ADDITONS, etc.** *(Attach Sheet if more Space is Required)*
____________________	____________________
____________________	____________________
____________________	____________________

DTRA or other GOVERNMENT CONTRACT NUMBER: ____________________

CERTIFICATION of NEED-TO-KNOW BY GOVERNMENT SPONSOR (if other than DTRA):

SPONSORING ORGANIZATION: ____________________

CONTRACTING OFFICER or REPRESENTATIVE: ____________________

SIGNATURE: ____________________

DEFENSE THREAT REDUCTION AGENCY
ATTN: IMCI
8725 John J. Kingman Road, MS-6201
Ft Belvoir, VA 22060-6201

DEFENSE THREAT REDUCTION AGENCY
ATTN: IMCI
8725 John J. Kingman Road, MS-6201
Ft Belvoir, VA 22060-6201

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED Technical 990401 - 000331

4. TITLE AND SUBTITLE Digital Communications in Fading and Jamming – COMLNK Users' Manual	5. FUNDING NUMBERS C – DTRA-01-99-C-0035 PE – 462D PR – AF TA – BK WU – DH000280
6. AUTHOR(s) Robert L. Bogusch	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Mission Research Corporation P. O. Box 2256 Atascadero, CA 93423-2256	8. PERFORMING ORGANIZATION REPORT NUMBER MRC-R-1607
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Threat Reduction Agency 8725 John J. Kingman Road, MS-6201 Ft. Belvoir, VA 22060-6201 TDANP/Schwartz	10. SPONSORING/MONITORING AGENCY REPORT NUMBER DTRA-TR-00-18

11. SUPPLEMENTARY NOTES
This work was sponsored by the Defense Threat Reduction Agency under RDT&E RMSS Code B 462D F210 AF BK BK 0028 A 25904D.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Digital communications systems that must operate successfully in the presence of signal propagation disturbances and radio frequency (RF) interference impose design requirements different from those for undisturbed channels. Without special attention, channel disturbances can degrade and possibly completely disrupt a communications link. Fortunately, the effects of RF propagation disturbances and interference, often referred to as signal fading and jamming, can be mitigated by careful design. This includes proper selection of modulation, coding, diversity, carrier tracking, synchronization, and signal acquisition techniques. The COMLNK computer simulation program was developed to facilitate the design of robust communications systems that must operate in adverse signal conditions. COMLNK provides a highly accurate numerical laboratory with which various design options can be quantitatively evaluated prior to commitment to hardware. The program is also useful in evaluation of existing systems, and it has proven to be invaluable in planning and conducting hardware tests. This document outlines the capabilities of the simulation and contains information useful in applying the code to the design, evaluation, and testing of digital communications systems.

14. SUBJECT TERMS

Signal Fading	Jamming	Scintillation	RFI
Modulation	Coding	Interleaving	Carrier Tracking
Signal Acquisition	Frequency Hopping	Spread Spectrum	Direct Sequence

15. NUMBER OF PAGES
183

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

PREFACE

The COMLNK code provides a detailed sampled-data simulation capability for the design, evaluation, and testing of digital communications systems that rely on radio frequency (RF) wave propagation. It has been used extensively over the past five years in support of various satellite systems and other wireless radio systems. Emphasis is placed on evaluation and mitigation of the effects of signal scintillation and RF interference, often referred to as signal fading and jamming. The code is applicable to a wide range of environments, from benign undisturbed signal propagation to highly disturbed frequency selective multipath conditions.

The deleterious effects of RF propagation disturbances and interference can be significantly alleviated by careful design, including proper choices of modulation, coding, diversity, and synchronization techniques. COMLNK was developed to facilitate the design of mitigated digital communications links and modems that must operate in a variety of signal conditions. It provides an accurate numerical laboratory with which design trades can be quantitatively evaluated prior to commitment to hardware. It is equally useful in evaluation of existing systems, and has proved to be indispensable in planning and conducting hardware tests.

COMLNK is available free of charge to qualified requestors. It is a stand-alone simulation code that runs on most Pentium-class desktop computers. Requests for the code should be directed to the Defense Threat Reduction Agency/TDANP, 6801 Telegraph Road, Alexandria, VA 22310-3398.

This document is the user's manual for COMLNK Version 5.3, which is the current release as of the time of writing (April 2000). The information in this document is essentially the same as contained in the on-line user's manual included with the code.

CONVERSION TABLE

Conversion factors for U.S. Customary to metric (SI) units of measurement.

MULTIPLY $\longrightarrow$ BY $\longrightarrow$ TO GET
TO GET $\longleftarrow$ BY $\longleftarrow$ DIVIDE

Angstrom	1.000 000 × E -10	meters (m)
atmosphere (normal)	1.013 25 × E +2	kilo pascal (kPa)
British thermal unit (thermochemical)	1.054 350 × E +3	joule (J)
calorie (thermochemical)	4.184 000	joule (J)
degree (angle)	1.745 329 × E -2	radian (rad)
degree Fahrenheit	$t_k = (t°f + 459.67)/1.8$	degree kelvin (K)
electron volt	1.602 19 × E -19	joule (J)
Erg	1.000 000 × E -7	joule (J)
erg/second	1.000 000 × E -7	watt (W)
Foot	3.048 000 × E -1	meter (m)
foot-pound-force	1.355 818	joule (J)
gallon (U.S. liquid)	3.785 412 × E -3	meter^3 (m^3)
Inch	2.540 000 × E -2	meter (m)
Kilotons	4.183	terajoules
Micron	1.000 000 × E -6	meter (m)
Mil	2.540 000 × E -5	meter (m)
Mile (international)	1.609 344 × E +3	meter (m)
Ounce	2.834 952 × E -2	kilogram (kg)
pound-force (lbs avoirdupois)	4.448 222	newton (N)
pound-force inch	1.129 848 × E -1	newton-meter (N-m)
pound-force/inch	1.751 268 × E +2	newton/meter (N/m)
pound-force/foot^2	4.788 026 × E -2	kilo pascal (kPa)
pound-force/inch^2 (psi)	6.894 757	kilo pascal (kPa)
pound-mass (lbm avoirdupois)	4.535 924 × E -1	kilogram (kg)
pound-mass-foot^2 (moment of inertia)	4.214 011 × E -2	kilogram-meter^2 (kg-m)^2
pound-mass/foot^3	1.601 846 × E +1	kilogram/meter^3 (kg/m^3)
Roentgen	2.579 760 × E -4	coulomb/kilogram (C/kg)
Shake	1.000 000 × E -8	second (s)
Slug	1.459 390 × E +1	kilogram (kg)
torr (mm Hg, 0° C)	1.333 22 × E -1	kilo pascal (kPa)

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Section Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Section Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

FIGURES

TABLES

SECTION 1
INTRODUCTION

1.1 BACKGROUND.

COMLNK was developed to facilitate the integration of performance and channel specifications, together with scintillation mitigation techniques, into the design and evaluation of robust digital communications systems (Bogusch, 1996). The program is intended to be useful throughout the life cycle of a digital communications system, including concept definition, specification, design, development, testing, and evaluation. It is intended for use by engineers at all stages from link architecture to waveform design.

When the idea of a PC-based program for design of digital communications in fading channels was first conceived, a database-driven program architecture was planned (Bogusch, 1989a). The original concept involved a set of transfer functions to characterize the response of each module in a communications link. It was planned that data for the transfer functions would be compiled from the results of numerous detailed simulations of various system configurations.

It was recognized at the outset that a database approach would limit program flexibility, in that only those system elements whose performance had already been simulated in detail could be included in any link configuration. Direct application of the underlying sampled-data digital simulation methods would provide a far more robust program, but personal computers in the late 1980's were not fast enough to support such an approach.

With the advent of advanced 32-bit processors, the PC has become sufficiently powerful to support the use of detailed simulation techniques. Consequently, in 1992 the development approach was changed to a simulation-based program architecture. At the same time, the decision was made to use only 32-bit code in the program to gain the full advantage of advanced microprocessors such as the Intel Pentium. This approach provides an infinitely more flexible program having exceptional accuracy and, hopefully, acceptable speed.

The resulting program, COMLNK, is an evolutionary design based on a large number of numerical simulations of communications systems that have been developed over the past two decades. Many of the techniques employed in these simulations have been described previously (Bogusch, 1989b; Bogusch, 1990; Bogusch and Michelet, 1993).

When first developed, most of the preceding simulation programs were aimed at specific applications for which the link layouts were reasonably well defined. Run-time changes primarily involved selection of design parameter values within a given link configuration. The link configuration itself tended to be hard-wired in each program and therefore not modifiable by means of user input. Consequently, as new system applications arose, a new program was generally constructed. Each program often made use of many of the same basic modules (encoders, interleavers, etc.), but their different layout required that the program structure be redesigned.

Two significant advances in simulation architecture have removed the constraints of these older simulation designs. First, the development of an interrupt-driven program architecture allows the layout of individual modules in the communications link to be varied dynamically at run time, providing an almost unlimited variety of user-selectable link configurations. Second, the application of dynamic memory allocation techniques enables data to be routed and shared efficiently between the various modules that comprise a digital communications link, regardless of their placement in the layout.

The significance of these architectural advances can be better understood when one considers the timing involved in the execution of the various processes in a digital communications link. A functional block diagram that is typically used to display the link layout may appear rather simple. Such a diagram shows that data flows through various modules in sequence as information is conveyed from source to sink. This view is deceptively simple, however. Even in synchronous communications, each process often operates at a different rate from those preceding or following. In addition, each process must await data at its input before it can provide output to the next process.

For example, a rate ½ error-correction encoder produces two output symbols for each input symbol, while the corresponding decoder must process two inputs for each output. A binary-to-octal converter must receive three inputs before it produces an output, while the complementary process does just the opposite. A demodulator may form many digital samples during each modulation symbol period with which to track the signal carrier and produce the output symbol metric, and thus executes at a different rate from other processes. More complicated timing relationships result when synchronization symbols or other data are multiplexed into, and then out of, the user data stream. The upshot of this is that the time-varying order in which the various modules in a link must be executed, and the times that they may be executed relative to other modules, can become quite complex even when the link layout is fixed. Now, allow the layout to be user-configurable in real time; the possible timing relationships become almost unbounded. It is the ability to efficiently accommodate these complexities that makes the program architecture important.

1.2 PROGRAM CAPABILITIES.

COMLNK employs sampled-data simulation models of the signal and data processing functions found in microprocessor-based digital communications equipment. Use of simulation techniques provides high accuracy and exceptional flexibility. The program architecture can be described as interrupt-driven with bus-oriented dynamic data exchange between modules.

Several advantages accrue from this architecture. The code is efficient both in terms of code size and execution speed. All code modules are reusable objects. The static and dynamic (state variable) data for each invocation of a given module is maintained internally as a separate dataset, with pointers passed to the module when it is executed. Thus, a link layout can utilize any combination of modules, and any module can be reused as many times as needed, without duplicating any code.

Quite complex link layouts can be constructed, and layouts can be easily edited through an interactive user interface. Modules can be inserted, deleted, and replaced in a layout via menu selection and cursor movement. Each module has a custom data menu that facilitates specification of design parameter values.

The sample data files distributed with the program provide examples of link configurations ranging from simple to fairly complex. The default layout, COMLNK.DAT, is an example of a relatively simple single-segment link. A screen shot of this layout is shown in Figure 1-1.

In the layout shown in Figure 1-1, the source module provides the transmitted user data bit stream from a specified message file or a random number generator. The resulting user data then passes through a cyclic redundancy check (CRC) block error detection encoder, a convolutional error correction encoder, a convolutional interleaver, and into a hop frame formatter multiplexer. The formatted data is modulated onto a frequency-hopped phase-shift keyed (FH/PSK) carrier. After

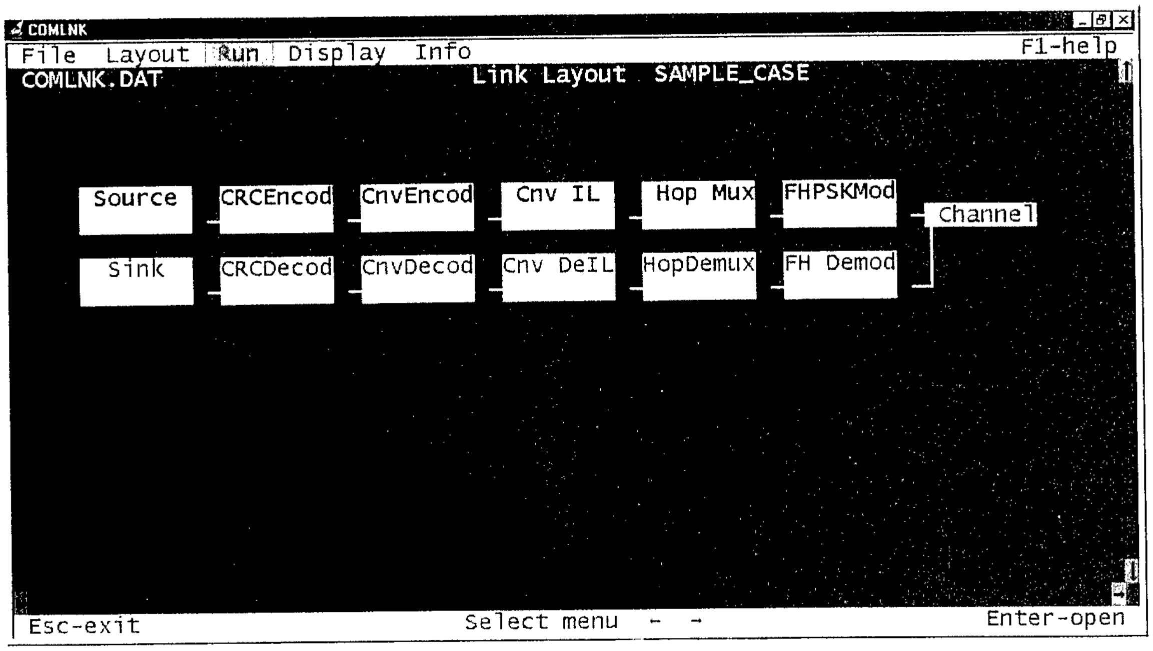


Figure 1-1. Example of a single-segment link (COMLNK.DAT).

propagating through a channel, which may exhibit any degree of disturbance, the received signal is demodulated, demultiplexed, deinterleaved, decoded and fed into the sink, where user bit errors and message errors are counted.

COMLNK accommodates a wide variety of applications. By editing the sample data files, or by building new layouts from scratch, many different types of digital communications systems ranging from simple to quite complex can be quickly constructed. The program has been extensively applied to a variety of satellite systems, including single links and multiple concatenated links. For example, end-to-end simulation of a network involving multiple terminals with numerous uplinks and downlinks has been accomplished.

Concatenated links are formed by adding additional link segments, each with its own processing functions, modems, and channels. A link with two segments, for example, may represent an uplink and downlink using a bent-pipe transponder or a partial or full processing satellite. A bent-pipe receives the uplink signal, frequency translates it and retransmits it on the downlink frequency. The data file BENTPIPE.DAT, whose layout is shown in Figure 1-2, is an example of a simple relay using a bent-pipe transponder.

Much more complicated links can be readily constructed. For example, a partial processing satellite demodulates the uplink data and then remodulates it onto the downlink carrier. A full processing satellite demodulates, deinterleaves, and decodes the uplink data and then reprocesses it through another chain prior to downlink transmission. The downlink processing functions may be the same as those for the uplink or they may be entirely different. In any case, each channel may exhibit any degree of disturbance.

The layout UPDOWN.DAT, shown in Figure 1-3, provides an example of a fully processed link. The layout UPXDOWN.DAT in Figure 1-4 shows that an uplink-crosslink-downlink configuration is formed by adding a third segment. More complex link layouts can be constructed with the introduction of additional link segments; see, for example, the layout contained in the file 2RELAYS.DAT (Figure 1-5). Note that some of these layouts exceed the screen dimensions. The layouts can be scrolled on the screen to display the entire link configuration.

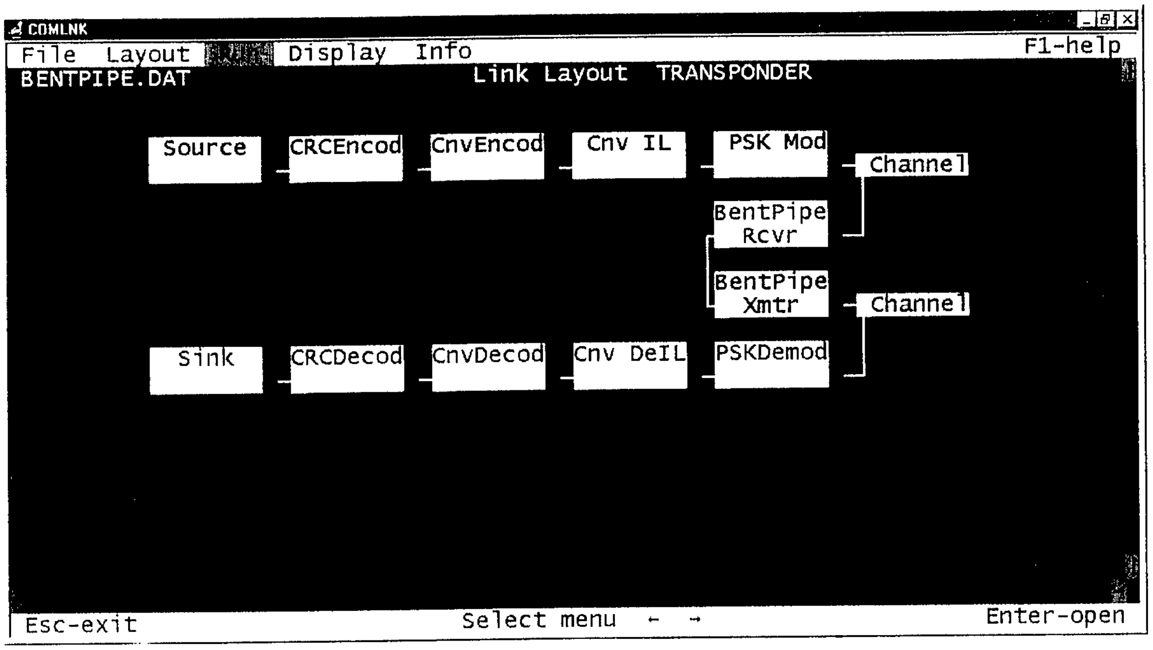


Figure 1-2. Example of a bent-pipe relay link (BENTPIPE.DAT).

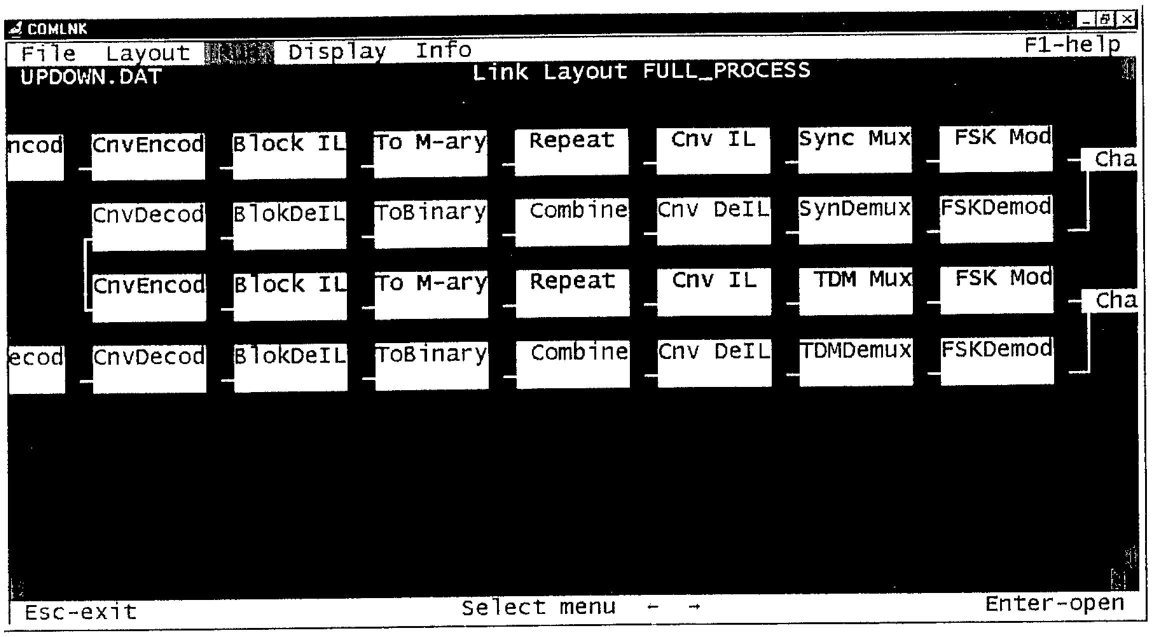


Figure 1-3. Example of a fully processed uplink-downlink (UPDOWN.DAT).

It is also possible to simultaneously simulate multiple independent links, providing an effective way of rapidly assessing the relative merits of competing designs. The file TWOLINKS.DAT, whose layout is shown in Figure 1-6, provides an example of this capability. These and many other example layouts are provided with the program.

Concatenated relay links and multiple independent links demonstrate the reuse of processing modules. Such layouts involve multiple instances of encoders and decoders, interleavers and deinterleavers, multiplexers and demultiplexers, modulators and demodulators, and channels.

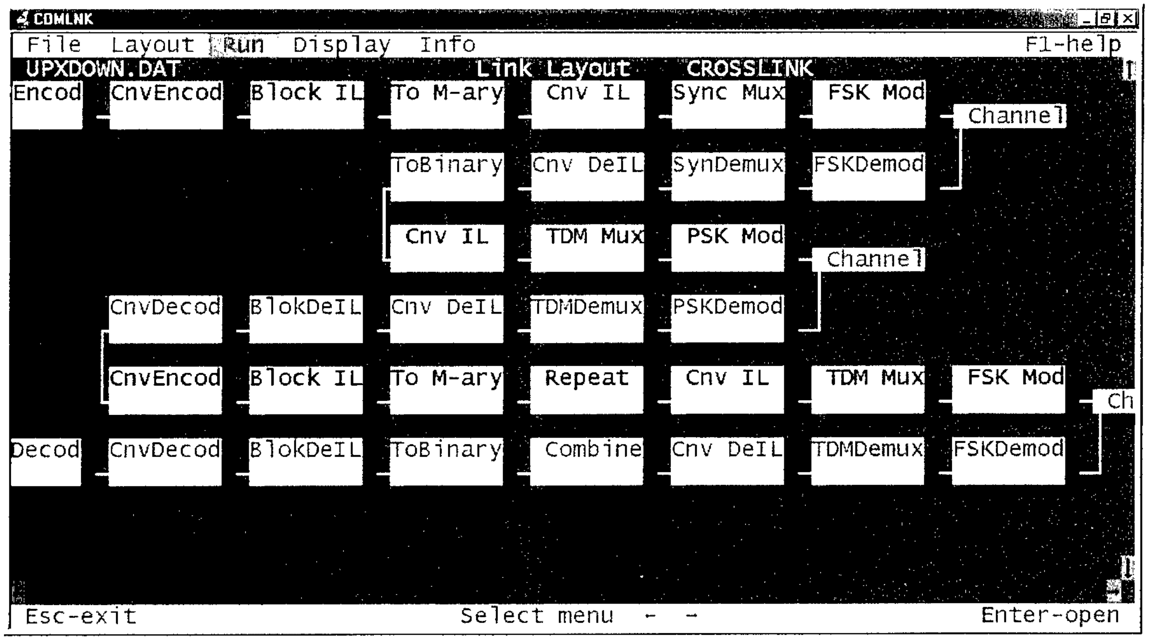


Figure 1-4. Example of an uplink-crosslink-downlink (UPXDOWN.DAT).

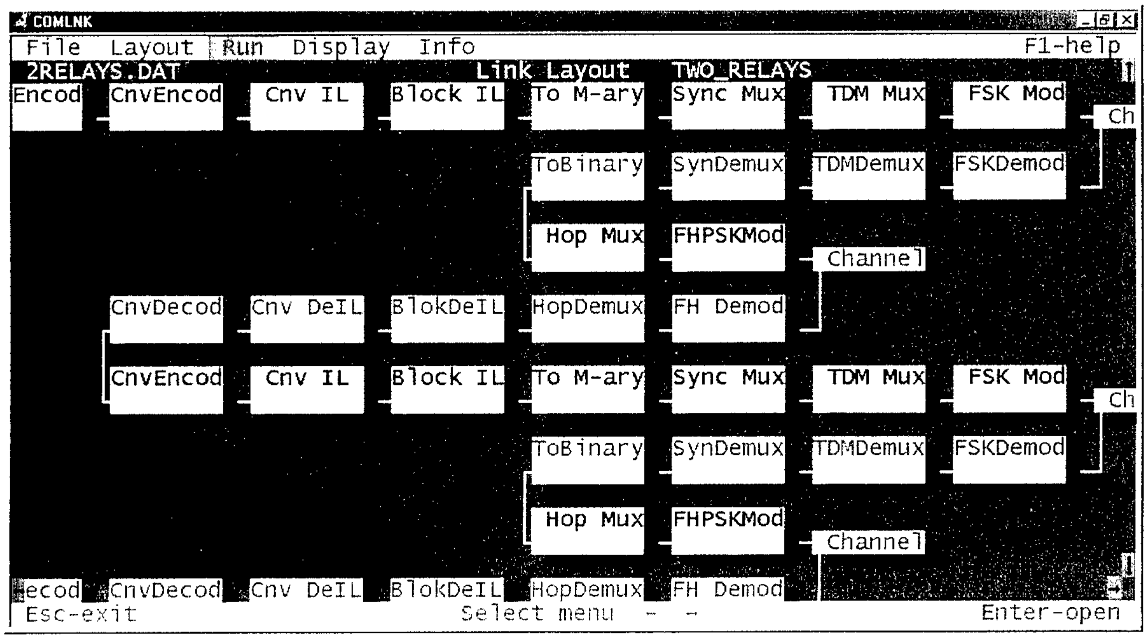


Figure 1-5. Example of a two-hop relay link (2RELAYS.DAT).

Each link segment may utilize different combinations of these and other processing functions. In addition, a variety of tracking loop designs can be selected for each modem. Each instance of a given type of module is a separate functional entity, utilizing the same executable code with different inputs and different state variable datasets.

COMLNK provides a fast, flexible and faithful simulation capability. It is designed to offer high accuracy together with minimum complexity and exceptional speed. When faced with

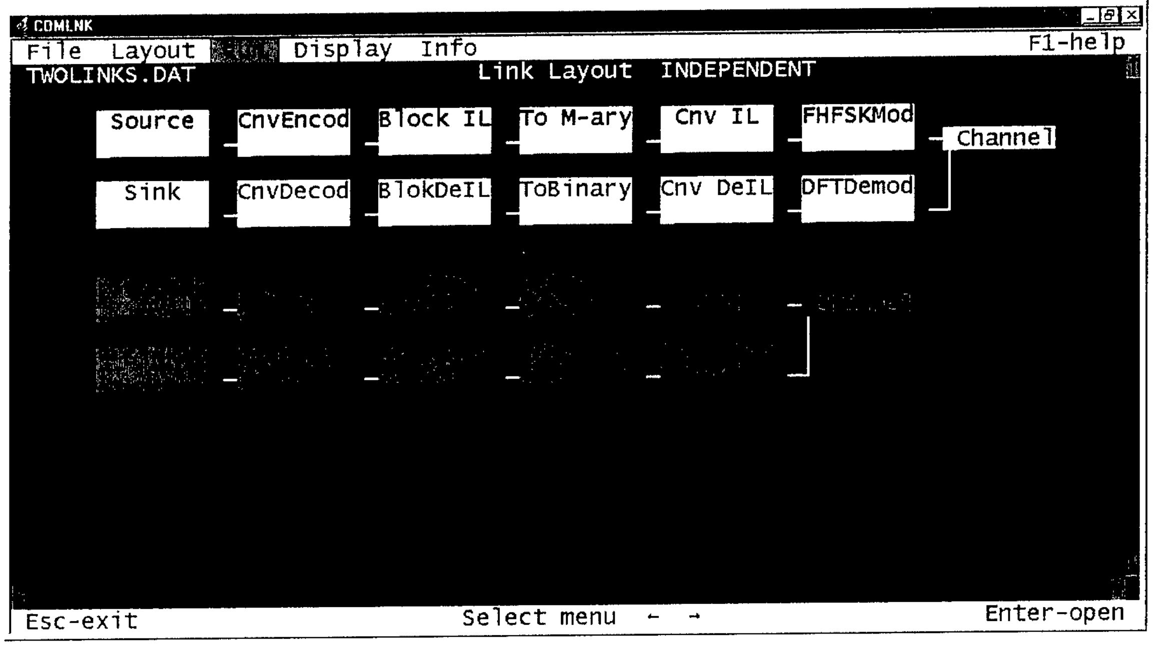


Figure 1-6. Example of two independent links (TWOLINKS.DAT).

architectural choices, program development was guided by faithfulness to the physical processes. The resulting program consists of about 100,000 lines of code, approximately 65 percent Fortran and 35 percent assembly language code.

1.3 SIMULATION MODULES.

The modules and capabilities currently available in COMLNK are summarized in the following paragraphs. COMLNK offers a variety of waveforms with a selection of modulation, coding, tracking, and synchronization functions. Direct-sequence and frequency-hopped spread-spectrum techniques are available. Independent-tone frequency-hopped FSK modulation is included, as well as more conventional FSK and PSK waveforms. Independent-tone FH/FSK is an antijam waveform that is resistant to multipath interference and hence may be suitable for low and medium data rate communications in fading and jamming environments.

COMLNK contains the Defense Nuclear Agency (now the Defense Threat Reduction Agency) models for flat and frequency selective fading channels. The channel decorrelation time and frequency selective bandwidth are specifiable parameters. Nonfading additive white Gaussian noise (AWGN) channels are also available.

COMLNK also includes the capability of simulating radio frequency interference (RFI) from hostile or friendly sources. A variety of jamming waveforms is available, including partial-band or full-band noise, partial-band or full-band tones, pulsed noise, pulsed tones, or linear-FM (chirp) pulses. The jammer path may exhibit fading and range dynamics, independent of the signal path, and the jammer may include a rotating antenna.

1.3.1 Summary of Available Modules.

A link layout can be constructed using any of the communications and channel modules provided with the code, and each module can be used any number of times. Currently available modules are briefly summarized below. More complete descriptions of each of these modules are given in Sections 4 and 5 of this document.

- ***Transmission Bit Source/Sink***
  - ASCII text (choice of 6-bit subset, 7-bit standard, or 8-bit extended ASCII codes)
  - Formatted or unformatted ASCII messages
  - Deterministic or pseudorandom binary data

- ***Error Detection Encoder/Decoder***
  - CRC with any parity generator polynomial, up to 32 bits wide
  - Any CRC block size, any shift register preset
  - Optional Hamming codes (conventional, extended, and specialized)

- ***Differential Encoder/Decoder***
  - Phase ambiguity resolution with PSK suppressed carrier tracking

- ***Error Correction Encoder/Decoder***
  - Any rate 1/N constraint length K code ($2 \leq N \leq 30$, $3 \leq K \leq 28$)
  - Any set of code generator coefficients (known good codes built-in)
  - Binary code or binary-to-M-ary code (matched to binary or M-ary modulation)
  - Catastrophic code generator detection
  - Hard or soft decision (1 to 8 bits) maximum-likelihood Viterbi decoder
  - Single or dual metrics with binary codes, M-ary metrics with M-ary codes
  - Optional linear feedback decoder with rate ½ K=10 systematic code

- ***Binary-to-M-ary/M-ary-to-Binary Converter***
  - Values of M up to 128 ($\log_2 M \leq 7$)
  - Hard or soft M-ary-to-binary conversion

- ***Symbol Repeater/Combiner***
  - Any repetition factor up to 2^{16} - 1 (65535)
  - Hard or soft M-ary symbol combining

- ***Interleaver/Deinterleaver***
  - Convolutional interleaver/deinterleaver
  - Block interleaver/deinterleaver
  - Optional block repeat/combine
  - Any interleaver depth and span
  - Deterministic or pseudorandom interleaving

- ***Multiplexer/Demultiplexer***
  - Frequency-hopped FSK (FH/FSK) sync symbol mux/demux, optional permutation
  - Frequency-hopped PSK (FH/PSK) reference mux/demux, optional permutation
  - Time-division mux/demux of multiple data channels
  - Uniform or nonuniform multiplexing, bit stuffing

- ***Modulator/Demodulator***
  - BPSK, QPSK, OQPSK (coherent or differentially coherent)
  - 2-ary to 128-ary FSK (conventional or independent tone)
  - Optional pseudonoise (PN) direct-sequence spread-spectrum with BPSK
  - Optional frequency hopping with BPSK, QPSK, and M-ary FSK
  - Discrete Fourier Transform (DFT) or tone-filter bank FSK demodulation
  - Automatic gain control (AGC), noise-based or signal-based

❑ ***Tracking/Synchronization***

- Delay-lock loop (DLL) time tracking
- Frequency-lock loop (FLL) carrier tracking
- Phase-lock loop (PLL) and composite FLL/PLL carrier loop in PSK demodulators
- Channel inversion detection and correction in OQPSK demodulators
- Sync-symbol and/or suppressed-carrier tracking in FSK demodulators
- Frequency ambiguity detection and correction in FSK demodulators
- Preamble frame acquisition with FSK (choice of algorithms and design options)
- Full cold-start acquisition available with PN/PSK (*e.g.* GPS L1 C/A acquisition)

❑ ***Channel***

- Additive white Gaussian noise (AWGN)
- Flat Rician/Rayleigh fading, f^{-4} Doppler spectrum, any Doppler spread (any τ_0)
- Frequency selective fading, f^{-6} Doppler spectrum, any delay spread (any f_0)
- Linear transponder (two-way fading, n-way fading)
- Statistically nonstationary (time-varying) channels
- Optional use of recorded channel response from field experiments
- Optional range and total electron content (TEC) dynamics profiles

❑ ***Jammer***

- Noise, CW tones, linear-FM (chirp) waveforms with any J/S
- Full-band or partial-band with any frequency offset
- Continuous or pulsed with any PRF and any duty cycle
- Any number of tones, any tone spacing for tone jammers
- Pulsed waveforms can be FH frequency followers
- Scintillation and/or range dynamics on jammer path
- Rotating jammer antenna with specifiable beamwidth
- Optional multiple jammers per receiver

The communications models in COMLNK have been developed during the course of numerous system simulation development efforts by the author and his colleagues (Bogusch, *et al.*, 1981; Bogusch, *et al.*, 1983; Bogusch, 1989b; Bogusch, 1990; Bogusch and Michelet, 1993). This work has been significantly aided by the published works of a large number of researchers. The relevant literature in this field is far too extensive to list in its entirety, but several works stand out in the author's memory (Cahn, *et al.*, 1977; Heller and Jacobs, 1971; Jacobs, 1974; Lindsey, 1972; Odenwalder, 1976; Oppenheim and Schafer, 1975; Papoulis, 1965; Proakis, 1966; Viterbi, 1966; Viterbi, 1967; Viterbi and Omura, 1979).

The channel models in COMLNK are software implementations of the Defense Threat Reduction Agency (DTRA) (formerly the Defense Special Weapons Agency [formerly the Defense Nuclear Agency]) channel model (Wittwer, 1979; Wittwer, 1980; Wittwer, 1982; Wittwer, 1993), which has been implemented in several hardware channel simulators. In addition to the standard flat (nonselective) Rayleigh fading channel, which is characterized by an f^{-4} Doppler spectrum, COMLNK includes the capability of smooth transition from nonfading AWGN, through Rician fading (Dana, 1993), to fully developed Rayleigh fading.

COMLNK also incorporates the DTRA dispersive channel model (frequency selective fading), which enables multipath effects to be treated for any waveform. The frequency selective fading channel is based on the turbulent model (Dana, 1991), wherein the channel impulse response is represented by narrowband Gaussian random sequences at a set of delay taps (Dana, *et al.*, 1995, Dana and Bogusch, 1998).

The tapped delay line implementation is basically the same as that incorporated in the DTRA Nuclear Effects Link Simulator (NELS II) (Cross. 1987) and in the recently developed DTRA Advanced Channel Simulator (ACS) (Dana. 1995). In our software implementation. the Gaussian sequences at each delay tap are generated by cascading three digital single-pole (RC-type) filters in the inphase and quadrature (I and Q) channels of the taps, giving rise to an f^{-6} Doppler spectrum (Dana, 1994).[1]

The software channel model in COMLNK enables a wider range of channel conditions to be simulated, with much less difficulty, than is possible using the NELS II hardware simulator. COMLNK encompasses the full range of channel conditions that can be tested using the ACS. Furthermore, since the channel model in COMLNK can be replicated as many times as needed in complex link layouts, a much wider range of system configurations can be evaluated in software than can be tested in hardware.

The resulting frequency selective channel impulse response is applied directly to produce intersymbol interference in links with large instantaneous signal bandwidths (high modulation rates). It is applied via discrete Fourier transform (DFT) techniques to links involving frequency hopping. The user need not be concerned with the details of the channel models. Once the channel frequency selective bandwidth and decorrelation time are specified together with the modulation rate and hopping bandwidth, all details involving the selection and application of the proper channel model are handled automatically by the program.

1.3.2 Application of Modules.

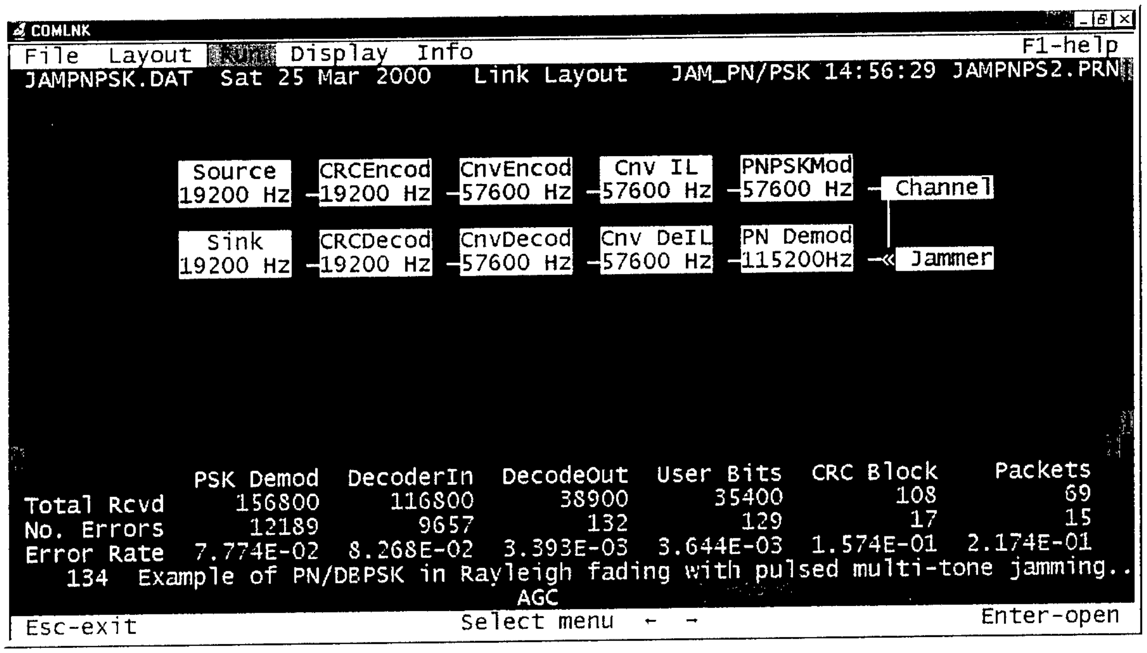
The modules summarized above may be installed in a link in any combination and in any order that makes sense. The program automatically analyzes the link layout and flags errors or inconsistencies for the user to correct prior to running the simulation. During this process, the link layout is displayed graphically on the monitor screen. Complex links that exceed the monitor dimensions can be scrolled horizontally and vertically on the screen so as to view all modules. This display capability remains available while the layout is running, as do essentially all of the menu functions in the interactive user interface.

When a layout is executing, a display of error counts at various points in the link is provided near the bottom of the screen. This display, which includes counts of demodulated bit errors, decoded bit errors, CRC block errors and packet errors, is updated frequently to permit the user to readily observe link performance. Packet error statistics are accumulated simultaneously for up to five packet sizes.

A line of received ASCII text or a hex error pattern is provided below the error-rate display. This line is usually updated on the screen whenever an error occurs. The display format and update interval are user options. An example of the error-rate display is provided in Figure 1-7, which is a screen dump obtained during a simulation run. Received character errors are displayed in red.

Graphical displays of received signal properties (carrier amplitude, phase, Doppler, delay) and receiver tracking loop operation (AGC gain, phase error, frequency error, delay error) are available by post-processing an optional plot output file. Figure 1-8 illustrates the type of graphical information that COMLNK can provide. This figure, from a COMLNK plot file, shows an example of a phase-lock loop pulling into phase and frequency lock following an initial frequency offset in the presence of signal scintillation. Also available in the plot file are cumulative and running average error rates, including demodulated bit error rate, user bit error rate, and block error rate.

[1] In flat fading, where a single delay tap is employed in the channel model, two RC filters are cascaded in the I channel and Q channel to provide an f^{-4} Doppler spectrum.



```
JAMPNPSK.DAT  Sat 25 Mar 2000    Link Layout    JAM_PN/PSK 14:56:29 JAMPNPS2.PRN

            PSK Demod   DecoderIn   DecodeOut   User Bits   CRC Block     Packets
Total Rcvd     156800      116800       38900       35400         108          69
No. Errors      12189        9657         132         129          17          15
Error Rate  7.774E-02   8.268E-02   3.393E-03   3.644E-03   1.574E-01   2.174E-01
   134  Example of PN/DBPSK in Rayleigh fading with pulsed multi-tone jamming..
```

Figure 1-7. Example of on-screen error-rate display.

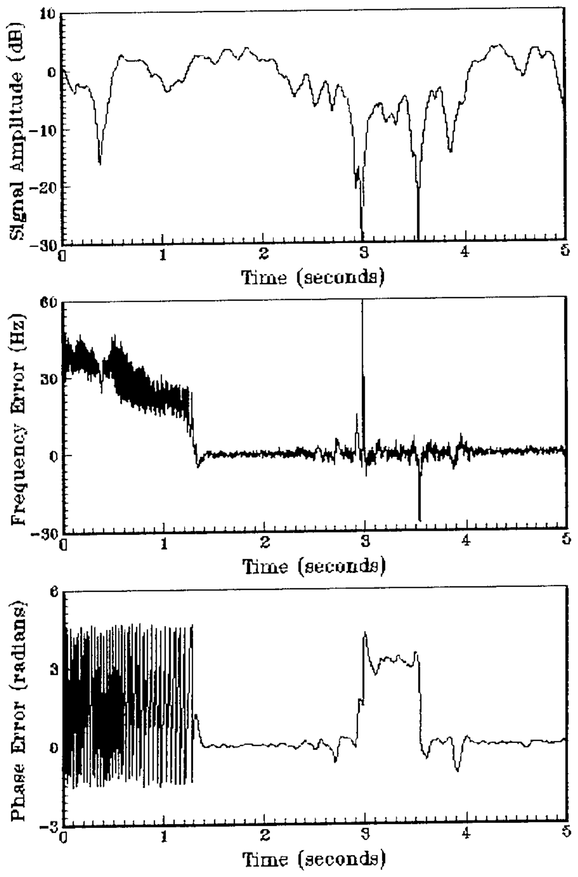


Figure 1-8. Example of PLL frequency and phase acquisition in a fading channel.

The interface responds when a key or mouse button is pressed, allowing the user to switch menu selections, modify the screen display, browse files, and edit layouts. Between keypresses or mouse clicks, the simulation continues to run at full speed. Very little overhead is involved in the interactive user interface. Therefore, work can proceed in the analysis of results and in the preparation of subsequent cases while the current job continues to execute in the background.

Default values are supplied for most design parameters, so that users of all levels of experience should have little difficulty in getting started. Numerous example layout data files are included with the program. Context sensitive help is available to assist with program operation, including menu navigation, layout editing, and design parameter specification. Additional information concerning the program, including the source code, is available through the information menu.

1.4 RUNNING TIME.

For speed and accuracy, all of the digital signal processing functions, including DFT filters and tracking loops, are programmed in pure integer arithmetic just as commonly done in actual equipment. All tracking loops are integer designs involving only multiplication and binary shift operations. The only run-time elements using floating-point operations are the channel models, the jammer modules, and the analog sections of the A/D converter routines. Time-intensive routines are hand-coded in assembly language for speed. A Fortran version of these routines is maintained for portability to other platforms.

COMLNK runs in 32-bit protected mode under DOS or under the DOS prompt in Windows 3.1, Windows 95/98, or Windows 2000 (but not older versions of Windows NT) on any PC that has a 32-bit x86 Intel-compatible processor with floating-point coprocessor (80386+80387, 486DX, Pentium, Pentium II, Pentium III). At least 8 MB of available RAM and a color VGA display or better are required.

Running time varies with the complexity of the link layout and the speed of the host PC. For typical single-link configurations involving CRC encoding, convolutional encoding, interleaving, multiplexing, FSK or PSK modulation, channel disturbances, sampled-data demodulation with carrier tracking, demultiplexing, deinterleaving, and soft-decision maximum-likelihood Viterbi decoding, the program runs at effective user bit rates of several kilobits per second or more on Pentium platforms. Thus, COMLNK runs sufficiently rapidly on most PCs that long simulations involving up to hundreds of millions of bits are practical, enabling bit error-rate measurements to be made down to the vicinity of 10^{-6}.

For example, the total run times for the set of demonstration cases (DEMO.DAT) and for the default COMLNK.DAT file are tabulated below for four PCs. These include a 400-MHz Pentium II machine, a 233-MHz Pentium, a 150-MHz Pentium, and a vintage 100-MHz 486DX4 notebook PC. All of the example print files on the distribution diskettes were produced with the 400-MHz Pentium II, and the resulting CPU times are provided at the end of each file.

Table 1-1. Run times on several PCs.

Data File	Pentium-400	Pentium-233	Pentium-150	486DX4-100
DEMO.DAT	1.1 min	2.2 min	4.2 min	12.6 min
COMLNK.DAT	3.7 sec	6.2 sec	10.5 sec	33.1 sec

Within a given chip family, run times scale approximately inversely with clock speed, although the cache size and floating-point unit (FPU) design also have an effect. Compared to the 486, the Pentium provides a speed advantage because of its dual-pipeline processor and improved FPU.

After scaling for clock speed, run times on a Pentium average around a factor of 1.5 to 2.0 less than on a 486. The Pentium II provides an additional speed advantage because of its larger cache and faster FPU.

COMLNK has been found by several users to be a good test of whether their computer is functioning properly. If a user's run times are significantly longer than the foregoing data would suggest, it is possible that the PC hardware may be incorrectly configured. Among other things, the CPU speed should be checked to make sure that it is set correctly. Most PCs enable the clock speed to be changed either manually or via setup software. On some machines, this may require checking that jumpers on the motherboard are positioned properly. One might also check the RAM speed and whether the secondary (L2) RAM cache is enabled.

Two users have reported that COMLNK run times revealed that their machines were incorrectly set up by the suppliers. Consequently, COMLNK has been employed as a benchmark to help evaluate competing machines during procurement. In this context it should be noted that COMLNK is CPU-bound and not I/O-bound.

1.5 SIMULATION ACCURACY.

Because COMLNK employs the same types of digital signal processing algorithms as used in actual microprocessor-based communications equipment and implements the algorithms using the same type of ROMable code, the program provides a faithful simulation capability. The code underwent beta testing throughout 1995 and has received extensive application to communications system design, evaluation, and testing.

Simulation modules in COMLNK have been validated through extensive testing and checking against theory, previous simulation results, and hardware test data. This validation procedure is an on-going process that utilizes new analysis and test results whenever they become available (Dana and Bogusch, 1998). In general, when the hardware design parameters are known, the program has been found to agree with test data to within about one decibel, and often to within a few tenths of a decibel.

Losses inherent in digital implementations, such as quantization loss and sampling loss, are included in COMLNK by virtue of its emulation of hardware digital signal processing. Losses that are not included are those associated with the analog sections of equipment, such as RF and IF plumbing loss and spectral distortion in mixers and analog filters.

In summary, when hardware tests are conducted on new equipment in a laboratory environment, results from COMLNK generally agree with test data to within the measurement accuracy. Indeed, COMLNK is often used to validate the test setup. In cases where analytical results are available, COMLNK exhibits essentially perfect agreement with theory. Here again, COMLNK has frequently been used to help verify new analytical results, especially those involving extensive numerical calculations.

However, any simulation must always be considered suspect until it is validated against other data sources, especially when used in new applications. The user is strongly encouraged to critically examine all results in light of other data and, whenever possible, compare the results to theory, other simulations, and test data.

Note that a simulation, even an excellent one, can only evaluate the design given as input. When existing equipment is evaluated, the user must determine that the specified design reflects the hardware. Hardware tests using channel simulators are necessary to determine whether equipment has actually been implemented in accordance with design specifications.

SECTION 2
USER INTERFACE

2.1 OVERVIEW.

The interactive user interface handles all communication between the user and the program. A menu bar at the top of the screen enables individual menus to be selected. A status bar at the bottom of the screen summarizes key commands that can be used for menu or function selection. "Hot keys" shown in red in the menu bar or in open menu lists provide a shortcut to menu selection or function selection. Optional control-key combinations enable frequently used functions to be accessed directly without opening the associated menu. The intent is to provide an intuitive interface with minimum complexity.

The available menus, together with the functions available under each menu when it is open, are listed in Table 2-1. All menu operations can be performed using only the keyboard. A mouse or other pointing device is supported but is not required by the interface.

Table 2-1. Menu functions.

File	Layout	Run	Display	Info
Open layout file	Insert module	eXecute layout	View this layout	Introduction
Edit this layout	Delete module	Analyze layout	sWap background	User interface
Save this layout	Replace module	Terminate link	How much longer	How do I ...
Browse any file	Module data	Program reset	picK error rates	Code modules
Quit program	Copy/cut/paste	rUn options	Graphics display	Source code

There are several ways in which these menus can be navigated. The traditional interface uses only the keyboard. Arrow keys can be used to move the menu highlight to the desired menu, and the Enter key (or the down arrow) then opens that menu. Once a menu is open, up and down arrow keys can be used to move the selection highlight to the desired function, and the Enter key then executes that function. The left and right arrow keys can be used to change menus while they are open so as to examine the functions available in each. The escape (Esc) key cancels a function or closes a menu. Thus, Enter always opens and Esc always exits, as their names imply.

Alternatively, hot keys can be used to open menus and execute functions. The capital letters in the foregoing list are displayed in red on the monitor screen. Red letters signify hot keys that open a menu if none is open, or execute a function if a menu is already open. Only keys that are currently active are displayed in red on the screen. When no menu is open, hot keys displayed in the top menu bar will open a menu. Once a menu is open, hot keys in the open menu list will execute a function. The effect is exactly the same as moving the highlight and pressing the Enter key. Again, the Esc key will cancel a function selection or close a menu.

Control (Ctrl) key combinations further simplify moving around the menus. By simultaneously pressing a Ctrl key together with a letter key, the function associated with that letter is immediately executed without having to first open the menu that contains the function. Except for the Info menu, control key assignments are the same as the hot keys displayed in the open menu lists. In most cases, the key is the first letter in the function name. For example, Ctrl-O opens a layout data file no matter where the user happens to be in the menu structure. Ctrl-S saves the current layout. Ctrl-B browses any file. Ctrl-V views a large layout by scrolling it around on the display. Ctrl-Q quits the program. Many other control key combinations are found by examining the capital letters in the menu functions in Table 2-1.

At any time, the F1 key displays a concise context-sensitive help message on the screen. The help messages are intended to assist in menu navigation. They also provide information concerning the operation of menu functions. When examining module data in the layout menu, F1 displays a brief description of each user-specifiable parameter in the data menus for the communications modules, channels, and jammers. In all cases, F1 is a toggle that enables or disables the screen display of help messages.

Context-sensitive help may be enabled automatically as the user navigates through the menus, depending on the amount of usage. The COMLNK help system keeps track of how much experience the user has with the program. Help is enabled frequently for new users. As use of the program continues, the automatic help feature gradually phases out. Help can always be manually invoked by pressing F1.

Once help is invoked either automatically or manually, help messages continue to be displayed as the user navigates through the menus and submenus. These messages are canceled when F1 is pressed again. Help messages are also canceled when Esc is pressed or when a run is initiated.

The COMLNK menu system remains active while a link layout is executing. Menus can be opened, help messages can be displayed, and functions can be executed while a simulation is running without significantly affecting the run time. For example, files can be opened and read in and layouts can be edited and saved while a run is in progress. Consequently, layouts can be constructed and the results of preceding runs can be examined while a simulation continues to run in the background.

If a mouse or other pointing device is available, it can be used to navigate the menus and control program operation. The left mouse button provides primary control in function selection. The right mouse button toggles help messages on or off in the same manner as the F1 key. Any combination of mouse and keyboard control can be used at any time. When a mouse is available, scroll arrows appear at the right side and bottom of the layout screen display. Clicking on the scroll arrows moves the layout around on the screen in the same manner as the Display menu View function.

Any of the menus can be opened at any time by clicking on the menu bar at the top of the screen. Clicking on an item in an open menu list executes that function. When a list of file names appears in a menu list, such as in the Open and Browse functions, a file can be opened by double-clicking its name or by moving the selection cursor with a single click and then clicking on the highlighted name. When keyboard commands are shown in the status bar at the bottom of the screen, clicking on any command will execute it exactly as if the indicated key had been pressed.

When a menu is open or an operation is awaiting user response, clicking anywhere in the layout area outside of the menu area closes the menu or backs out of the operation in the same manner as pressing the Esc key. The exception to this rule arises when an open submenu contains the Cancel option; in that case, the user must click on Cancel or one of the other submenu options to exit that submenu.

Any time that a layout is visible on the screen, clicking on a module in the layout opens its data menu. It may be necessary to click on the module more than once if another function is active in order to back out of that function and open the module data menu. Once a data menu is open, scroll arrows within the menu can be used to move the highlight up or down. The bottom scroll arrow may be obscured by a submenu symbol, but it remains active. The highlight can also be moved by clicking on a data item, causing the highlight to move toward that item. Clicking on a highlighted item opens the data entry field. After typing in the data, it is entered by pressing the Enter key or by clicking on the Enter command at the bottom of the screen.

2.2 FILE MENU.

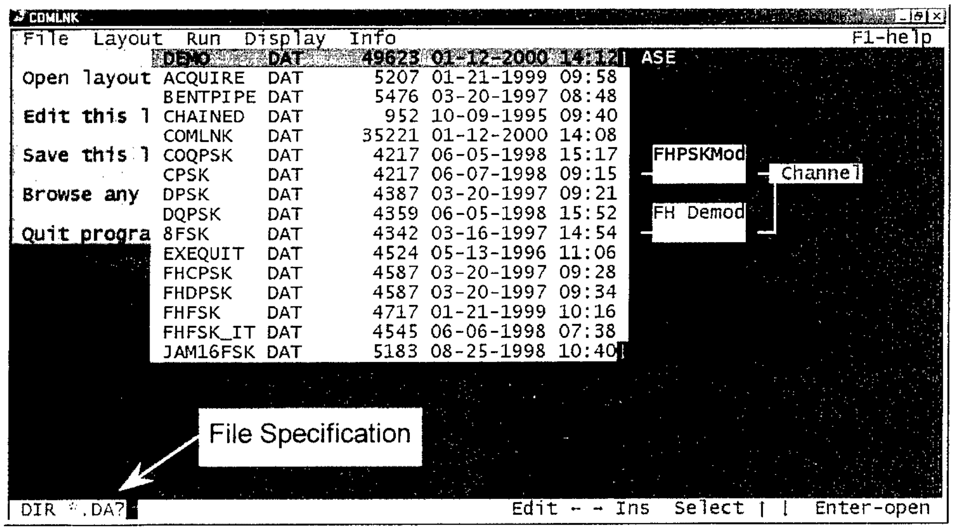
The File menu provides the functions needed to open layout data files, save layout files, browse any file, and terminate the program. The File menu is automatically entered when COMLNK is started in order to open a layout file that is displayed when the program title screen goes away. A file for this initial layout may be specified on the DOS command line when COMLNK is invoked, or the previous data file may be loaded when the program starts.

Thus, when the program loads it looks for an input data file in two places. It first checks the command line to see if a filename, which may optionally contain the DOS path, was entered. If so, the program attempts to open that file. If no filename was entered, the program attempts to open the layout data file that was in use when the previous session ended. Hence, one can automatically resume where previously left off. If there was no previous session, the program attempts to open the default data file, COMLNK.DAT, in the current directory. In any case, the name of the file that the program uses is shown at the upper left of the screen.

If a blank screen with the message "File not found" appears when the title screen goes away, the program was unable to find the initial layout data file in the specified or current directory. To get a layout on display, or to replace a layout on display with another one, open the File menu and select the function Open layout file.

2.2.1 Open Layout File.

Selecting this function produces a list of files on the screen, if there are any files that match the file specification. The file specification is shown at the lower left of the screen (see Figure 2-1). This specification has the same format as the DOS DIR command. It may contain part or all of a directory path as well as a filename. The filename may contain the DOS wildcard characters (* and ?) and hence provides a filter to display a list of files that match the specification.



```
DEMO      DAT     49623 01-12-2000 14:12
ACQUIRE   DAT      5207 01-21-1999 09:58
BENTPIPE  DAT      5476 03-20-1997 08:48
CHAINED   DAT       952 10-09-1995 09:40
COMLNK    DAT     35221 01-12-2000 14:08
COQPSK    DAT      4217 06-05-1998 15:17
CPSK      DAT      4217 06-07-1998 09:15
DPSK      DAT      4387 03-20-1997 09:21
DQPSK     DAT      4359 06-05-1998 15:52
8FSK      DAT      4342 03-16-1997 14:54
EXEQUIT   DAT      4524 05-13-1996 11:06
FHCPSK    DAT      4587 03-20-1997 09:28
FHDPSK    DAT      4587 03-20-1997 09:34
FHFSK     DAT      4717 01-21-1999 10:16
FHFSK_IT  DAT      4545 06-06-1998 07:38
JAM16FSK  DAT      5183 08-25-1998 10:40

DIR *.DA?
```

Figure 2-1. File specification for open layout file.

Initially, the file specification is set to *.DA?. This causes a list of all files in the current directory having an extension beginning with DA to be displayed. Consequently, when Open layout file is initially selected, a list of data files will appear if the current directory is the one in which files were copied from the distribution diskettes. If no such list appears, a message is displayed suggesting that the file specification should be edited.

The file specification can be edited to access any directory on any active disk drive in the host machine. For example, a specification of *.* displays all files in the current directory, including entries for the parent directory and any subdirectories that may be present. A file specification of *. displays only directories and files without an extension. The directory can be easily changed by selecting a directory entry from the list of files, thereby moving up or down the directory tree. For example, selecting the directory entry <..> enters the parent directory, which is one level closer to the root directory on the current disk drive. Directories are clearly indicated in the list of files, just as they are when the DIR command is entered at the DOS prompt.

Files on a different drive can be accessed by including the drive specification at the beginning of the file specification. For example, a file specification of A:*.* displays all files and directory entries on a floppy disk. Just be sure to insert a properly formatted diskette before selecting a floppy drive.

Alternatively, a path can be added to the file specification to access a different directory. For example, a file specification of ..*.* lists all files in the directory preceding the current one. A specification of *.* lists files in the root directory. These types of specifications access other directories without changing directories. This is the recommended procedure when a simulation is running. If the directory is changed during a run the simulation will continue unaffected, but a blank layout will appear when it is finished. This occurs because temporary files created at the start of the run no longer reside in the current directory once it is changed.

Any name and extension can be assigned to layout data files. Although the example layout files on the distribution diskette all have a .DAT extension, there is no requirement to do so. Different file extensions may be used to organize files between different projects, for example.

To obtain a listing of files, or to modify one on display, edit the file specification using the standard editing keys (backspace, delete, left arrow, right arrow). The Insert key toggles between insert and replace modes, and the cursor changes color to indicate the selection. Use Ctrl-Delete to delete all characters from the cursor position to the end of the file specification. The file specification can contain any drive and path acceptable to DOS. Files and directories that match the specification, if any, will be listed in a window with a selection highlight. Up and down arrow keys scroll through the list. The Page Up (PgUp) and Page Down (PgDn) keys move one group at a time. Ctrl-Home and Ctrl-End go to the top and bottom of the list.

Once a list of data files or directories appears, move the highlight to the one wanted and press Enter. Alternatively, type enough of the name until the desired entry is highlighted and then press Enter. A selection can also be made by double-clicking an entry or by moving the highlight with a single click and then clicking the highlighted item. If a directory is selected, it becomes the current directory and a new list of files and directories matching the specification will appear. If a file is selected, it will be read in and can be edited in the Layout menu or executed in the Run menu.

If the message "No layout in file" appears, a file has been selected that does not contain a link layout and therefore cannot be a valid input data file. If the message "Not an input file" appears, the file is binary and thus is not an input file. In either case, the Browse function (described below) can be used to examine the file if desired. In any event, because the selection is not an input data file, one should either choose a different file using the Open function, or use the Layout menu to create a new link layout.

2.2.2 Edit This Layout.

This function transfers to the Layout menu so that the link layout on display can be edited, or a new layout can be built. If no simulation is currently running, the effect is the same as opening the Layout menu.

The primary purpose of this function is to facilitate editing a layout that is currently running. Because a running layout cannot be changed, this function makes a copy that can be edited. The original layout continues to run unchanged in the background.

2.2.3 Save This Layout.

This function saves the layout on display to a disk file. The save layout function contains the functionality of both the "save" and "save as" functions found in other programs.

When this function is selected, a filename is suggested. The suggested filename is the name of the original input file, if any, or the name under which this layout was saved most recently. To save the layout under the suggested name, select OK. A message will appear confirming that the file has been saved.

To save the layout under a different name, select Rename. Edit the filename at the bottom of the screen using the usual editing keys (backspace, delete, left arrow, right arrow). Insert toggles between insert and replace modes, and the cursor changes color accordingly. Ctrl-Delete deletes all characters from the cursor position to the end of the filename string. Any filename acceptable to DOS, with one to eight name characters and up to three extension characters, can be specified. Press Enter when editing is finished, or press Esc to cancel the edit.

Note that layout data files may have any extension, not just .DAT. Different file extensions can be used to organize data among several projects, for example. When the desired filename has been entered, select OK to save the layout under the new name. A confirming message will then appear.

Do not be concerned about overwriting a file. The program automatically saves an existing file having the same name by renaming it with a slightly modified extension. Even prior backups are saved by this procedure. For example, say that a layout is saved under the name MYLINK.LAY and that a file with that name already exists. The existing file is renamed MYLINK.LA0 and the new file becomes MYLINK.LAY. If this name is used again, the existing MYLINK.LAY is renamed MYLINK.LA1 and the new file again becomes MYLINK.LAY.

Therefore, the most recently saved file generally has the specified name, with older files having a series of extensions that reflect the order in which they were created. No data is ever lost by saving a layout, no matter how many times the same name is used. The last character of the extension is varied over the range 0-9 and A-Z. After these 36 characters are used, subsequent saves cause the program to modify the name of the new file. In such cases, the new name is formed by changing the last character of the specified name.

If the filename is unacceptable to DOS, the program substitutes another name. The name LAYOUT.DAT is used if the name was blank. The name COMLNK00.DAT is used if the name contained other characters unacceptable to DOS. In any case, the name that is actually used is displayed when the layout is saved.

If a layout has been changed and not saved, the program automatically saves it before quitting using the name AUTOSAVE.DAT. Again, backups are saved by renaming them with a modified extension. Thus, if a file with the name AUTOSAVE.DAT already exists, it is renamed AUTOSAVE.DA0, and so on. The objective is to prevent the accidental loss of data. Unwanted files can easily be deleted using the DOS or Windows delete commands. Files can also be deleted while browsing them with the following function.

2.2.4 Browse Any File.

This function enables any file, regardless of the file type, to be viewed on the screen. When this function is selected, a list of files that match the file specification appears. The file specification, shown at the lower left of the screen (see Figure 2-1), has the same format as the DOS DIR command. It may contain a directory path and a filename. The filename may contain the DOS wildcard characters (`*` and `?`) and provides a filter to display a list of matching files.

Initially, the browse file specification is set to `*.PR?`. This causes a list of all files in the current directory having an extension starting with PR to be displayed. Consequently, when the browse function is first used, a list of print output files will appear if the current directory is the one in which print files were copied from the distribution diskettes. If no such list appears, a message is displayed which suggests that the file specification should be edited.

The file specification can be edited to access any directory on any active drive in the host machine. All files and directories in the current directory, or any subset of them, can be displayed. Directories can be changed by selecting a directory entry, thereby moving up or down the directory tree. Alternatively, a path can be added to access other directories without changing directories (recommended when a simulation is running). Refer to the foregoing discussion of Open layout file for information on editing the file specification and changing directories.

Files that match the specification, if any, will be listed in a window with a selection highlight. Up and down arrow keys scroll through the list. PgUp and PgDn move one group at a time. Ctrl-Home and Ctrl-End go to the top and bottom of the list. Move the highlight to the name of the desired file, or type enough of its name until it is highlighted, and press Enter. A file can also be selected by double-clicking its name. The file is then displayed in a large window on the screen.

Any file of any type can be browsed. This includes layout data files, print output files, and other ASCII files. Even binary files, including program executable files and object module files, can be browsed. The display switches automatically between a text format for ASCII files and a hexadecimal format for binary files. The display can be switched manually between the two formats using the F2 key. Hence, any file can be browsed in either an ASCII text format or a hexadecimal dump format.

In either format, files can be browsed line-by-line using the up and down arrow keys, or page-by-page with the PgUp and PgDn keys. The Home and End keys jump to the top or bottom of the file. As noted, the F2 key toggles between ASCII and hexadecimal display formats.

In ASCII format, the left and right arrow keys scroll the display horizontally. Ctrl-Tab toggles between expansion and display of tab characters, with the tab expansion incremented or decremented by the Ctrl-GrayPlus and Ctrl-GrayMinus key combinations using the numeric keypad.

Files can be browsed when a layout is running without disturbing the run. A file can also be deleted if it is no longer needed. To delete a file, press the Delete key while browsing that file. A prompt will request confirmation before the file is deleted. Then, when OK is selected, the file is "erased" in the usual DOS manner.[1]

[1] Deleted files no longer appear in a DOS or Windows directory listing, but remain on the disk until overwritten. Files deleted under DOS are not moved to the Windows recycle bin.

2.2.5 Quit Program.

This function is used to exit COMLNK and return to the operating system. If the program is running from the DOS command line, this function will return to the DOS prompt. If the program was started using a desktop icon installed under Windows, it may be necessary to close the COMLNK window. Refer to the help topic on how to install COMLNK on the Windows desktop.

If a layout has been edited or any module data has been changed and the modified layout has not yet been saved, it will be saved automatically under the name AUTOSAVE.DAT when the program quits.

When a simulation is run, the results are contained in print output files having a .PRN extension. Any binary plot data files created by the run will have a .BIN extension. Any received message text files created by the run will have an .RCV extension. The name of each of these files is taken from the name of the layout data file that was run.

2.3 LAYOUT MENU.

The Layout menu provides the functions needed to create and edit link layouts. The layout from any file can be edited, or a completely new layout can be constructed using any combination of available modules. Files can be opened and read in and layouts can be edited and saved while a run is in progress.

A previously constructed layout can be read in from a data file using the File menu and then edited in the Layout menu. A brand new layout can be built from scratch using the functions in the Layout menu. The Insert module function provides a list of available modules that can be incorporated in link layouts.

Build or edit a layout using the Insert, Delete, Replace, Module data, and Copy/cut/paste functions. A given type of module can be used as many times as desired in a layout. For example, it is a simple matter to design a link with concatenated coding. Each instance of a module can be given the same or different design data. Concatenated links and multiple links can be constructed. The data files on the distribution diskette provide a few examples of the many types of links that can be built.

Once work has started on a layout, it convenient to use control keys to rapidly switch between Insert (Ctrl-I), Delete (Ctrl-D), Replace (Ctrl-R), Module data (Ctrl-M), and Copy/cut/paste (Ctrl-C) functions. The active function is displayed at the bottom of the screen, and the module selection highlight color is changed to correspond to the active function. A mouse can also be used to switch functions by clicking Layout in the top menu bar.

When a layout is finished, the Analyze function in the Run menu can be used to check the design for inconsistencies or inappropriate module placement.

Using the menu functions is by far the easiest way to edit and save layout data files. However, these files can also be edited off-line using any text editor that can read and write plain ASCII files. The COMLNK.DAT and DEMO.DAT files contain information for users who wish to edit layout files off-line. Be sure to save layout files as ASCII text files if a word processor is used to edit them off-line.

Note again that existing layouts can be read in and edited, new layouts can be constructed, and displayed layouts can be saved to disk while a simulation is running. Working on a layout does not interfere with or impede the progress of a run. The run executes in the background while work proceeds in the foreground. The foreground and background jobs can be swapped at any time using functions available in the Copy/cut/paste submenu and in the Display menu.

2.3.1 Insert Module.

The Insert module function is used to construct a new layout or to add modules to an existing layout. Because work can proceed on either of two layouts, display the one wanted via the swap background function in the Display menu. Ctrl-W provides a shortcut to swap background.

When the Insert module function is selected, a list of available modules appears on the screen. By moving a selection highlight and an insertion point highlight, the selected module can be inserted into the layout at a specified point as illustrated in Figure 2-2.

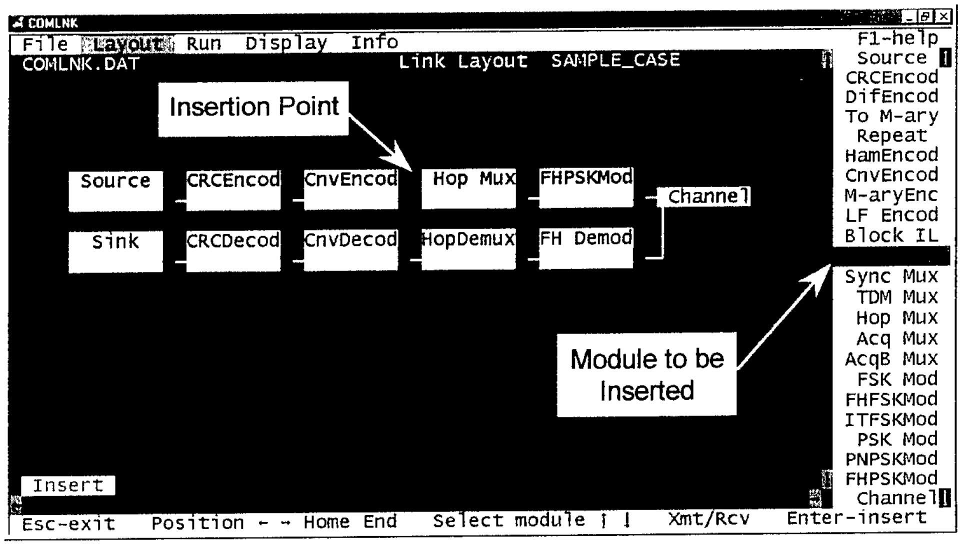


Figure 2-2. Insert module function in layout menu.

Select the module to be inserted with the up and down arrow keys, and select the insertion point with the left and right arrow or Home and End keys. The left arrow moves toward the source module at the start of the layout; the right arrow moves toward the sink module at the end. The Home key moves immediately to the first source module. The End key moves to the end of each link segment (normally a channel or jammer module) and then to the last sink module in the layout.

The insertion point can also be selected by clicking on the module preceding the point at which the new module is to be inserted in the layout. The module to be inserted can be selected by clicking on it in the list. When both the module and the insertion point have been selected, press Enter or click the selected module to insert it in the layout.

Modules are always inserted in pairs having the complementary transmit and receive functions. For example, when a source module is inserted, the corresponding sink module is automatically inserted as well. Insertion of an encoder causes both the encoder and corresponding decoder to be installed, and so on.

The module list can be toggled between the corresponding transmit and receive functions by means of X and R keys. For instance, either a modulator or a demodulator can be inserted, with the other member of the pair automatically included. In general, it is good practice to insert transmit modules in the layout, as the corresponding receive modules are automatically inserted.

If a transmit module is inserted in the receive chain, or a receive module inserted in the transmit chain, a concatenation is formed where partially or fully processed received data is fed into another transmit chain. Insertion of a source module starts a new link that is completely independent of all other links in the layout. These concatenation and multiple link features enable construction of quite complex links and networks.

Insertion of a module at a concatenation point often leads to more than one way that the complementary receive or transmit module can be linked into the layout. The user is informed of such ambiguities when they arise and given the option of choosing another linkage. In such cases, the spacebar cycles through the installation options until the desired layout configuration appears on screen.

After a module has been inserted, the Delete key provides an "undo" function and deletes it. This is a convenient way to correct an insertion error. If several modules have been inserted consecutively, repeated presses of the Delete key provide a multi-level undo capability. This capability remains in effect until all modules inserted consecutively have been deleted, or until the insertion point is moved or the insert function is terminated.

Press the Esc key to exit the insert function. Alternatively, functions can be rapidly switched using control key combinations.

2.3.2 Delete Module.

The Delete module function is used to remove one or more modules from a layout. Modules are always deleted in pairs to remove the complementary transmit and receive functions. For example, when an interleaver module is deleted, the corresponding deinterleaver module is also deleted.

Select the module to be deleted with the arrow keys or by clicking it with a mouse. The left arrow moves toward the source module at the start of the layout. The right arrow moves toward the sink module at the end. The up and down arrows move between the complementary transmit and receive modules. The Home key moves immediately to the first source module. The End key moves to the end of each link segment (normally a channel or jammer module) and then to the last sink module. When the module to be removed is highlighted, press Enter or click Delete to delete it. The corresponding transmit/receive module pair will vanish.

After a module pair has been deleted, the Insert key provides an "undo" function and reinserts the pair of modules where they had been as long as the highlight is not moved prior to the undo. This is a convenient way of correcting a deletion error.

When a module pair is deleted and reinserted at a concatenation point, there may be more than one way in which the transmit or receive module can be linked into the layout. The user is informed of such ambiguities when they arise in the undo process and given the option of choosing another linkage. In such cases, the spacebar cycles through the reinstallation options until the proper layout configuration appears on screen.

The deletion undo feature provides a multi-level undo capability. If several modules are deleted in sequence, successive presses of the Insert key restore the modules in their original sequence.

This capability can be demonstrated using the example layouts distributed with the program. For example, read in the default layout COMLNK.DAT. Open the Layout menu and select the Delete function. Move the highlight to the channel module at the end of the link (the End key facilitates this). Press the Enter key repeatedly until all modules in the layout have been deleted. Then press the Insert key repeatedly; all of the modules will be restored in their original order, leaving the layout unchanged.

This undo capability remains in effect until all consecutively deleted modules have been reinserted, or until the module highlight is moved or the delete function is terminated.

Deleted modules are placed in the copy/cut buffer. Modules in this buffer can be inserted into a layout using the paste function, which is available under the Copy/cut/paste submenu.

The delete function can be exited at any time by pressing Esc, or functions can be rapidly switched using control key combinations.

2.3.3 Replace Module.

The Replace module function is used to change one or more modules in a layout to a different type. This function is equivalent to a deletion followed by an insertion.

When the Replace module function is selected, a list of available modules appears on the screen, similar to that illustrated in Figure 2-2. By moving two selection highlights, one in the module list and the other in the layout, one module can replace the other.

Select the module to be replaced with the left and right arrow keys, and select its replacement with the up and down arrow keys. The left arrow moves toward the source module at the start of the layout; the right arrow moves toward the sink module at the end. The Home key moves immediately to the first source module. The End key moves to the end of each link segment (normally a channel or jammer module) and then to the last sink module.

Alternatively, the module to be replaced can be selected by clicking on it in the layout, and its replacement selected by clicking in the module list. When both the old and new modules have been selected, press Enter or click the new module in the list to effect the replacement.

Modules are always replaced in pairs having the complementary transmit and receive functions. For example, when an encoder module is replaced, the corresponding decoder module is automatically replaced as well.

The module list can be toggled between the corresponding transmit and receive functions by means of X and R keys. In general, it is good practice to always replace transmit modules with other transmit modules. If similar functions are replaced (transmit with transmit or receive with receive) a similar layout will result. If dissimilar modules are replaced (transmit with receive or vice-versa), a concatenation is formed where partially or fully processed received data is fed into another transmit chain.

When a module pair is replaced at a concatenation point, there may be more than one way in which the transmit or receive module can be linked into the layout. The user is informed of such ambiguities when they arise and given the option of choosing another linkage. In such cases, the spacebar cycles through the replacement options until the desired layout configuration appears on screen.

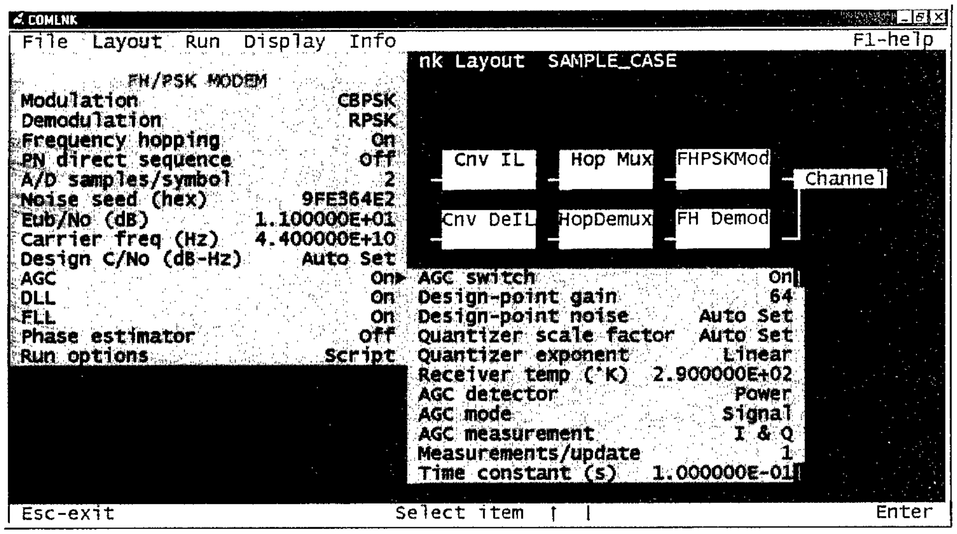
The Esc key exits the replace function. Alternatively, functions can be rapidly switched using control key combinations.

2.3.4 Module Data.

The Module data function is used to specify or examine module design parameter values. Parameter values can be examined in any layout, whether it is running or not. Values can be changed only when the layout is not running. To change values in a running layout, first make a copy of it using the Edit function in the File menu. Design parameters then can be changed in the copy.

A module selection highlight appears when the Module data function is selected. This highlight can be moved with the cursor keys to each module in the layout. The left arrow key moves toward the source module at the start of the layout. The right arrow key moves toward the sink module at the end. The up and down arrows move between the complementary transmit and receive modules. The Home key moves immediately to the first source module. The End key moves to the end of each link segment (normally a channel or jammer module) and then to the last sink module in the layout.

When a module is highlighted, the Enter key opens a data menu and displays the parameter values for the highlighted module as shown in Figure 2-3. The data menu for any module in the layout can also be opened simply by clicking on it. Data menus are always given for transmit/ receive module pairs. A highlight in the data menu can be moved to each parameter. The F1 key provides a brief description of the highlighted parameter and indicates the acceptable range of values and any default values.



```
        FH/PSK MODEM
Modulation                  CBPSK
Demodulation                 RPSK
Frequency hopping              On
PN direct sequence            Off
A/D samples/symbol              2
Noise seed (hex)         9FE364E2
Eub/No (dB)          1.100000E+01
Carrier freq (Hz)    4.400000E+10
Design C/No (dB-Hz)      Auto Set
AGC                           On▸ AGC switch                         On
DLL                            On Design-point gain                  64
FLL                            On Design-point noise           Auto Set
Phase estimator               Off Quantizer scale factor       Auto Set
Run options                Script Quantizer exponent             Linear
                                  Receiver temp (°K)       2.900000E+02
                                  AGC detector                    Power
                                  AGC mode                       Signal
                                  AGC measurement                 I & Q
                                  Measurements/update                 1
                                  Time constant (s)        1.000000E-01
```

Figure 2-3. Modem data menu with open AGC submenu.

A black arrow at the right of a highlighted data field indicates that a data submenu is available. Either the Enter key or the right arrow key opens the submenu. The Esc key or the left arrow key closes the submenu. With a data menu open and a parameter highlighted, the Enter key opens a data entry field at the bottom of the screen in which a new parameter value can be typed. Data can be edited as it is typed using the normal editing keys (backspace, delete, left arrow, right arrow). Insert toggles between insert and replace modes, and the cursor color changes accordingly. Ctrl-Delete deletes all characters from the cursor position to the end of the data entry field.

When data entry is complete, signified by pressing Enter, the entry is checked for overflow or invalid characters. If it passes this test, it is then checked to make sure that it is within the proper range for the highlighted parameter. If not, it is limited to that range (or else rejected outright if it failed the first test). The result replaces the previous value in the data menu. When the user is unsure of an acceptable range for the parameter, help is available by pressing F1. In addition, entering zero in a numeric data field will cause a default value to be substituted in cases where zero is not an appropriate value.

Some parameters have a predefined set of allowable values (usually two or three, occasionally more). In such cases, the Enter key displays the next value in the sequence. Press F1 for a brief description of the parameters.

As noted above, data for any module can be examined but not changed when a layout is running. This is useful to verify that the values are those intended, and to observe default values that the program may set for certain parameters at run time. Moreover, when running channels with time-dependent parameters, the changing values can be observed in the channel data menu. Simply move the data menu cursor to force the program to refresh the channel menu display.

2.3.5 Copy/Cut/Paste.

The Copy/cut/paste menu item provides a submenu that contains several other editing functions. These include functions to copy modules, cut modules, and paste modules into the layout from the copy/cut buffer. There is also a function to flush the copy/cut buffer, a function to delete all modules in the current layout, and a function to swap the foreground and background layouts.

2.3.5.1 Copy Modules. The copy function enables a copy of one or more modules in the layout, together with the associated design parameter values, to be placed in the copy/cut buffer. The modules remain in the layout, which is unchanged by this operation. For example, parts of a layout under construction can be copied to form multiple link segments. Part or all of a running layout can be copied to enable some or all of it to be pasted into a background layout.

Once the copy function is selected, a module highlight appears. Use the arrow keys to highlight the first or last module in a sequence to be copied. Press Enter to fix this point, whereupon the complementary transmit and receive module pair is highlighted. Then use the arrow keys to highlight the sequence of module pairs to be copied. Press Esc if necessary to start over and select a new initial point. Once the desired sequence of modules is highlighted, press Enter to perform the copy operation. A copy of the modules is then placed in the copy/cut buffer for subsequent pasting into this or another layout.

2.3.5.2 Cut Modules. The cut function operates similarly to the copy function, with modules and their design values placed in the copy/cut buffer. The major difference from copy is that cut removes modules from the layout as they are placed in the buffer. Hence, the cut function is similar to the delete function described earlier, except that cut provides a multi-module delete capability.

Once the cut function is selected, a module highlight appears. Use the arrow keys to highlight the first or last module in a sequence to be cut. Press Enter to fix this point, whereupon the complementary transmit and receive module pair is highlighted. Then use the arrow keys to highlight the sequence of module pairs to be cut. Press Esc if necessary to start over and select a new initial point. Once the desired sequence of modules is highlighted, press Enter to perform the cut operation. The modules are then removed from the layout and placed in the copy/cut buffer for subsequent pasting.

2.3.5.3 Paste In Copy/Cut. The paste function enables modules in the copy/cut buffer to be inserted in a layout. Choose the foreground or background layout and then select paste. A list of modules in the buffer is displayed at the right side of the screen, similar to that shown in Figure 2-2. Use the up and down arrow keys to select the module to be inserted in the layout. Use the left and right arrow keys to highlight the insertion point. Then press Enter to paste the selected module into the layout at the desired point. Pasted modules are removed from the buffer as they are used. To reuse modules, copy them after pasting.

The paste function is similar to the insert function described earlier, except that modules are selected from the copy/cut buffer rather than from the main module list and therefore retain previously specified design parameter values. Modules are always pasted in pairs having the complementary transmit and receive functions. The X and R keys toggle the list of modules in the copy/cut buffer between these functions. The Delete key provides an "undo" function after one or more modules have been pasted.

Pasting a module at a concatenation point often leads to more than one way in which the complementary receive or transmit module can be linked into the layout. The user is informed of such ambiguities when they arise and given the option of choosing another linkage. The spacebar cycles through the options.

2.3.5.4 Flush Copy Buffer. The flush buffer function empties the copy/cut buffer, thus providing a clean start for subsequent copy/cut/paste operations.

2.3.5.5 Delete This Layout. The delete layout function provides a clean start to construct a completely new layout. The layout window is cleared, and the deleted modules are placed in the copy/cut buffer.

2.3.5.6 Swap Background. The swap background function exchanges the background and foreground layouts. Neither layout is changed. This function facilitates work on two layouts, such as copying modules from one layout to another. This swap function is the same as that available in the Display menu.

2.4 RUN MENU.

The Run menu provides the functions needed to initiate and terminate simulation runs. A layout on display can be run or analyzed, a run can be terminated, the program can be reset, and certain run options can be examined.

2.4.1 Execute Layout.

A layout is run, *i. e.*, simulation of the link layout is initiated, by selecting the execute layout function. This function automatically performs the analysis described below before launching the run. Thus, when the execute layout function is selected to initiate simulation of the link layout on display, the Analyze layout function is automatically selected first.

If no errors are detected in analysis, the run begins immediately. A detected error causes a warning to be issued. The user then has an opportunity to correct the error, or he may choose to ignore the warning and proceed with the run.

Warnings that may have been ignored in a previous analysis of this layout will be displayed again before the run begins. This provides the user with another opportunity to either correct a problem or ignore the warning. Some errors are so severe that they cannot be ignored. In such cases, the warning will continue to be displayed and the run will not commence.

Use the Module data function in the Layout menu to examine the module data prior to running a simulation. Alternatively, some of the most commonly changed data can be examined using the run options menu item described below. When ready to proceed, Ctrl-X provides a shortcut to initiate a run.

2.4.2 Analyze Layout.

This function checks the layout on display for inappropriate module placement or other layout errors. The process examines the sequence of modules and some of the associated design values for consistency. Errors or inconsistencies are noted on the screen.

For example, installing an interleaver in an unhopped coherent PSK link causes a warning to be issued. The warning in this case informs the user that deinterleaving the demodulated channel symbols prior to resolving the phase ambiguities that are inherent in suppressed-carrier coherent PSK demodulation is an invitation to catastrophic link failure. This is true in any channel in which the carrier phase tracking loop may suffer cycle slipping.

Thus, if analysis of the layout detects what appears to be a problem, a warning appears on the screen. If the user agrees with the assessment, the program transfers to the Layout menu where the problem can be corrected. Alternatively, the user may choose to ignore the warning and proceed with the analysis. Except in the case of very severe errors, the program respects the user's decision to ignore warnings.

If warnings are ignored and the layout is subsequently run, the same warnings will be redisplayed to provide another opportunity to examine the layout prior to execution. If the warnings are again ignored and the link performance seems strange, the run can be manually terminated as described below.

The analysis function is most useful for preliminary checking when one has finished building or editing a layout. A new layout can be analyzed while a simulation is running in the background, without disturbing the current run.

While the analysis function is able to detect many errors in a layout, it is not foolproof. The user is strongly advised to carefully check the link layout and all design parameter values before making extensive runs.

It should be noted that completing this analysis without warnings does not guarantee good link performance. The designer of a communications link should experiment with the design to find one that exhibits satisfactory performance over the range of channel characteristics that pertain to the application at hand. COMLNK does not attempt to automatically find an optimum link design. Rather, it attempts to make the designer's efforts in this regard more efficient.

2.4.3 Terminate Link.

This function allows the user to terminate all links in a layout that is currently executing. When running a layout from an input file that contains two or more cases, the next case will automatically start when the current run is terminated. If the file does not contain another case, or if the layout was constructed on-line, the program will await further instructions when the run terminates.

A run will terminate automatically once it attains the specified number of user bits or user bit errors, whichever is reached first. These maximum values are specified in the source/sink data menu. These values can be examined using the run options menu item below. Alternatively, go to the Layout menu, select Module data, highlight the source or sink module, then open the data menu and examine the simulation duration values.

If a running layout contains more than one link, each link will automatically terminate once it attains the maximum number of user bits or bit errors for that link. The data menu for each source module shows the simulation duration values.

If it is desired to terminate the run prematurely, all links in the layout are simultaneously terminated by selecting the Terminate link function and responding in the affirmative when asked to confirm the action. A print file containing a summary of all of the input parameters, together with the results of the run, is always produced.

If a running layout contains more than one link, it is possible to prematurely terminate one link at a time. This is accomplished by entering the number of the link to be terminated using the Alt-keypad procedure. For example, if only the first link in a layout is to be terminated, hold down the Alt key and press and release the number 1 on the numeric keypad; then release the Alt key. Link 1 will then terminate, leaving the other links in the layout running. Note that the numeric keypad must be used for this procedure; the number keys at the top of the keyboard will not work here. Note also that the keyboard NumLock light must be lit.

Again, note that links will terminate automatically when the specified number of user bits or bit errors is reached. While a run is progressing, a display of error counts at various points in the link is updated frequently on the screen so that link performance can be continuously monitored. The final error counts appear in the print file produced at the end of the run. The program will provide an estimated time to finish the run if desired; go to the Display menu and select How much longer.

2.4.4 Program Reset.

This function resets the program to its initial state. This may be helpful if an out-of-memory warning is encountered. This function is also convenient to completely terminate a long scripted run that calls other files. It can be used any time that it is desired to start from a completely clean slate. In any event, work in progress is automatically saved to disk and thus is not lost.

2.4.5 Run Options.

This menu item opens a submenu that facilitates examination of parameters that control run length, plot output, text output, channel properties, and jammer characteristics. The submenu functions provide convenient access to some of the same data that can be examined using the Module data function in the Layout menu. The functions in this submenu are listed below.

2.4.5.1 Run Length Settings. These values are part of the source module data. The run length is set by the maximum number of user bits, or the maximum number of bit errors. A run automatically terminates when either value is reached. Any number of bits up to $2^{31} - 1$ (2,147,483,647) can be run, although this may result in some intermediate counters saturating at their maximum values. Any number of errors can be specified. When running a fading channel, the run length should encompass at least several thousand decorrelation times for reasonable statistical significance.

Selecting this function moves to the first source module in the layout and opens the data menu at the proper point to examine the run length settings. If the layout contains more than one link, the run lengths can be set individually for each. The settings for the other links can be examined while in the Module data function in the Layout menu. Move the module selection highlight to the other source modules and open the data menu to examine the run length settings. When finished examining the data, Ctrl-U provides a shortcut back to the run options submenu.

2.4.5.2 Plot Output Options. These values are part of the modem module data. A plot data file is optionally written during a simulation run depending on the plot on and off times, specified in the modem run options submenu. If the plot off time is later than the plot on time, a plot file is written if at least some of the plot interval overlaps the simulation time period, which begins at a specified start time. The resulting plot data file contains time histories of selected quantities such as received signal characteristics, tracking loop operation, and demodulator performance.

Selecting this function moves to the first modem in the layout and opens the data menu at the proper point to examine the plot times and plot data selection. If the layout has more than one modem, plot files can be turned on or off and plot data can be selected for each modem individually. Plot settings of any modem can be examined while in the Module data function in the Layout menu. Move the module selection highlight to the other modulator or demodulator modules and open the data menu to examine the run options. When finished examining the data, Ctrl-U provides a shortcut back to the run options submenu.

2.4.5.3 Text Output Options. These values are part of the source module data. The received message text (for an ASCII source) or error syndrome (for a hexadecimal or random source) is optionally written to an output file, depending on a switch setting in the source/sink module display options submenu. The three-position switch can be set so that no file is written, only received lines containing errors are written, or the entire reception is written. These three options can be selected individually for the on-screen display and the output text file.

Selecting this function moves to the first source module in the layout and opens the data menu at the proper point to examine the output file switch. If the layout has more than one link, this switch can be set individually for each. The settings for other links can be examined while in the Module data function in the Layout menu. Move the module selection highlight to the other source modules and open the data menu to examine the display options. When finished examining the data, Ctrl-U provides a shortcut back to the run options submenu.

2.4.5.4 Channel Parameters. These values constitute the data for the channel module. They include the S_4 scintillation index, which specifies the intensity of signal amplitude fading, the scintillation decorrelation time τ_o, which specifies the signal fading rate or Doppler spread, and the frequency selective bandwidth f_o, which specifies the channel coherence bandwidth or delay spread. Other channel parameters include phase noise, total electron content (TEC) variations, and range dynamics.

The channel can be statistically stationary, or it can have time-varying statistics whose parameters are obtained from a specified file. Both of these options utilize an on-line channel model that generates specific realizations of the random channel impulse response function. The channel can also be described by digital recordings of the inphase (I) and quadrature (Q) response. Recorded I-Q channel response data files may be obtained from any source including field experiments.

Selecting this function from the run options submenu moves to the first channel in the layout and opens the data menu. If the layout has more than one channel, the data for each can be examined while in the Module data function in the Layout menu. Move the module highlight to subsequent channel modules (the End key facilitates this) and open the data menu to examine the channel parameters. When finished examining the data, Ctrl-U provides a shortcut back to the run options submenu.

2.4.5.5 Jammer Parameters. These values constitute the data for the jammer or radio frequency interference module. They include the waveform, frequency band, and strength of the interference. Other jammer parameters include propagation path characteristics, range dynamics, and antenna rotation.

The jammer can radiate continuous or pulsed noise, one or more continuous or pulsed tones, or linear-FM (chirp) pulses. Any of the waveforms can be full or partial band. Pulsed jammers can be frequency followers if the system employs frequency hopping. The jammer can have a rotating antenna on a moving vehicle and can have a fading propagation path independent of the signal path.

Selecting this function from the run options submenu moves to the first jammer in the layout and opens the data menu. If the layout has more than one jammer, the data for each can be examined while in the Module data function in the Layout menu. Move the module highlight to subsequent jammer modules (the End key facilitates this) and open the data menu to look at the jammer data. When finished examining the jammer parameters, Ctrl-U provides a shortcut back to the run options submenu.

2.4.5.6 Get New Layout. This function simply invokes the Open layout file function in the File menu, where a new input file can be read in.

2.5 DISPLAY MENU.

The Display menu provides functions that affect the display of link layouts and simulation results. A layout on display can be scrolled to view all modules. The foreground and background layouts can be swapped. The time remaining in a current simulation run can be estimated. The set of displayed error rates in a current simulation run can be selected. It is planned for the display to be switched into graphics mode to show modem operation and channel parameters. Except for the graphics function, which is planned for a future release, all functions are currently operational although some are available only when a simulation is executing.

2.5.1 View This Layout.

This function activates the view mode, which allows the layout to be scrolled around on the screen using the keyboard. This enables the entire configuration of a large layout that extends beyond the confines of the display window to be viewed. Note that scrolling a layout does not affect the operation or speed of a simulation that may be running.

A layout can be scrolled at any time, whether it is running or not. Ctrl-V provides a shortcut to this function. Once in view mode, the layout can be scrolled using the cursor movement keys. These include the arrow keys, Home key, and End key.

The arrow keys move the layout up, down, left, or right one row or column at a time. End moves the layout so that the end of each link, usually a channel or jammer module, is on display. Home moves the layout so that the first module, usually a source module, is on display. Either Home or End can find a layout that has scrolled off the screen.

The PgUp and PgDn keys swap the foreground and background layouts in view mode. This performs the same function as the swap background function described below. Thus, these keys toggle the display between the primary and secondary layouts. The layout title bar at the top of the screen under the menu bar indicates which layout is on display. Press Esc to exit view mode.

When a mouse is available, it is not necessary to enter view mode to scroll a layout around on the screen. The layout can be scrolled at any time using scroll buttons at the right side and bottom of the display. Clicking these buttons moves the layout on the screen one row or one column at a time, exactly like the arrow keys do in view mode. Holding the left mouse button down while it is pointing to one of the scroll buttons produces continuous scrolling.

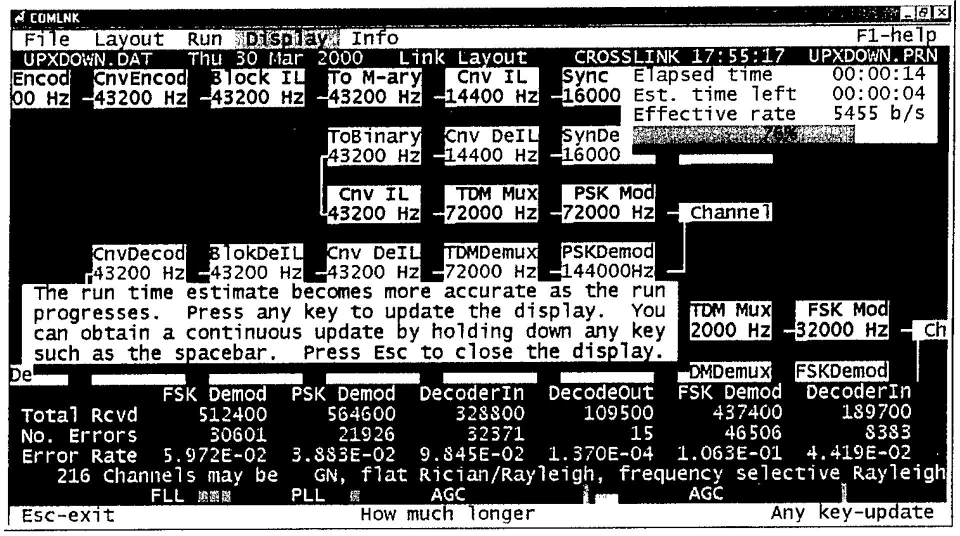
2.5.2 Swap Background.

This function swaps the foreground and background layouts. If a second layout has not yet been built or opened, a blank layout will appear. The Layout menu can then be used to construct a new layout from scratch. Alternatively, another layout can be read in from a data file. Either way, it is easy to work on two layouts at once, even when one is running.

Foreground and background layouts can be swapped at any time. Ctrl-W provides a shortcut to this function. The layout title bar at the top of the screen under the menu bar shows the name of the original input file, if any, and indicates the time at which the layout was run. This helps to identify which layout is currently on display.

2.5.3 How Much Longer.

This function can be used to obtain an estimate of how much more time will be required to complete a simulation run in progress. Selecting this function brings up a display of the current elapsed run time, an estimate of the time required to complete the run, the effective run speed in user bits per second, and the current completion percentage. Figure 2-4 shows an example of this display.



```
COMLNK
 File  Layout  Run  Display  Info                                      F1-help
 UPXDOWN.DAT   Thu 30 Mar 2000   Link Layout    CROSSLINK 17:55:17  UPXDOWN.PRN
Encod  CnvEncod   Block IL   To M-ary   Cnv IL     Sync  Elapsed time     00:00:14
00 Hz -43200 Hz -43200 Hz -43200 Hz -14400 Hz -16000  Est. time left   00:00:04
                                                      Effective rate   5455 b/s
                           ToBinary   Cnv DeIL   SynDe
                           43200 Hz -14400 Hz -16000

                           Cnv IL     TDM Mux    PSK Mod
                           43200 Hz -72000 Hz -72000 Hz - Channel

        CnvDecod  BlokDeIL  Cnv DeIL  TDMDemux  PSKDemod
        43200 Hz -43200 Hz -43200 Hz -72000 Hz -144000Hz
 The run time estimate becomes more accurate as the run
 progresses.  Press any key to update the display.  You   TDM Mux   FSK Mod
 can obtain a continuous update by holding down any key  2000 Hz -32000 Hz - Ch
 such as the spacebar.  Press Esc to close the display.
De                                                        DMDemux  FSKDemod
                FSK Demod  PSK Demod  DecoderIn  DecodeOut  FSK Demod  DecoderIn
 Total Rcvd        512400     564600     328800     109500     437400     189700
 No. Errors         30601      21926      32371         15      46506       8383
 Error Rate     5.972E-02  3.883E-02  9.845E-02  1.370E-04  1.063E-01  4.419E-02
   216 Channels may be   GN, flat Rician/Rayleigh, frequency selective Rayleigh
                FLL          PLL          AGC                  AGC
 Esc-exit                         How much longer                  Any key-update
```

Figure 2-4. Elapsed time screen display.

The estimated time to completion becomes more accurate as the run progresses. The display can be updated during the run by pressing any key. A continuous update can be obtained by holding a key down, such as the spacebar.

2.5.4 Pick Error Rates.

This function can be used to select the set of error-rate measurements that are updated on the screen during a simulation run. When this function is selected, a list of all of the error counts being accumulated in the current run is displayed. The ones currently being updated on the screen are marked in the list. The up and down arrow keys enable each measurement to be highlighted. The Enter key or the spacebar or the mouse can be used to select or deselect the highlighted measurement.

Any set of error-rate measurements, up to a maximum of six at any one time, can be displayed. If six measurements are on display and it is desired to pick a different set, first deselect one or more of the current measurements. Then select the desired ones from the list.

Alternatively, scroll the error-rate display horizontally to view all of the measurements in groups of six. The left arrow key moves the selection of the first displayed measurement toward the top of the list. The right arrow moves toward the end of the list. The error-rate display can be scrolled completely off the screen with the right arrow key. The left arrow key will scroll it back on screen.

No matter which set of error-rate measurements is displayed, all measurements in the list are summarized in the print output file produced at the end of the run. Changing the on-screen display has absolutely no effect on the error-rate measurements.

2.5.5 Graphics Display.

This function is not yet operational. When implemented, it is planned that this function will enable the user to select one or more graphical displays of channel conditions and modem operation while a simulation is running. A large number of quantities are available for plotting by post-processing an optional plot data output file. See the help topics on how to display results graphically and how to plot simulation results.

2.6 INFO MENU.

The Info menu provides on-line documentation of COMLNK and contains essentially the same material as that presented in this report.[1] Information provided by this menu supplements the help messages that are displayed by pressing the F1 key. Whereas the help messages offer concise context-sensitive assistance with the currently active menu function, the documentation provided by the Info menu gives a broader description of the program. This is essentially an on-line user manual.

The information provided by this menu is stored in a set of files contained on the distribution diskettes. These files have names in the format COMLNK#x.INF, where x is a letter denoting the specific type of information in the file. The information files are basically standard ASCII text files with a special header record that the program uses to access and display the file contents. A printed copy of these information files can be obtained by loading them into a text editor or word processor. Simply delete the header record and send the file to a printer. Be sure to select the IBM PC extended ASCII symbol set (see your printer manual). This is necessary to obtain proper printing of mathematical and drawing characters contained in the files. The files can be printed as is, or they can be formatted as much as desired. In any case, be sure to make a backup copy of the original information (.INF) files first; these files must not be changed.

If the message "System file COMLNK#x.INF not found" appears on the screen when the Info menu is used, the indicated file is not in the DOS directory where the program expects to find it. All system files (COMLNK.EXE and COMLNK#x.INF) should be in the same directory. Copy the information files from the distribution diskettes into the directory containing the COMLNK.EXE executable file.

If the message "System file COMLNK#x.INF corrupted" appears on the screen when the Info menu is used, the program has detected some problem with the indicated file. This can occur if the information file has been altered using a text editor or word processor, which can damage the file header record. If this message is displayed, simply copy the original files from the distribution diskettes (or from a backup copy) into the directory containing the COMLNK.EXE file.

2.6.1 Introduction.

This menu item provides the introductory section of the program documentation. We urge all users to read this material to gain overall familiarity with the program. The up and down arrow keys allow the user to scroll through the list of introductory topics. As the list is scrolled, the first few lines of each section are displayed in a small window. The entire section can be viewed by pressing the Enter key or clicking the list of topics.

Once a section has been opened for viewing, it can be browsed line-by-line with the up and down arrow keys, or page-by-page with the PgUp and PgDn keys. The Home and End keys jump to the

[1] Indeed, this report was created by formatting and editing the information menu text files as described above, and then adding a few illustrations.

top or bottom of the section. The Esc key returns to the list of introductory topics. These commands can also be executed by clicking them at the bottom of the screen.

2.6.2 User Interface.

This menu item provides a discussion of the interactive user interface. As before, the up and down arrow keys scroll through the list of menus, with the first few lines of each menu description displayed in a small window.

The entire description of each menu item can be viewed by highlighting it in the list of contents and pressing Enter or by clicking it with a mouse. Once an item has been opened for viewing, it can be browsed line-by-line with the up and down arrow keys, or page-by-page with the PgUp and PgDn keys. The Home and End keys jump to the top or bottom of the item. Esc returns to the list of menu items.

2.6.3 How Do I.

This menu item provides a list of help topics that are intended to answer any questions that the user may have concerning program operation. The up and down arrow keys scroll through the list of help topics. As this list is scrolled, the first few lines of each topic are displayed in a small window. Press Enter or click the topics list to read the entire topic, then press or click Esc to return to the list of topics.

2.6.4 Code Modules.

This menu item provides a technical discussion of the communications modules and channel models in COMLNK. A submenu allows the user to choose between these two categories. Once information on channels or communications has been selected, a table of contents appears. Use the up and down arrow keys to move through this list of technical sections while the first few lines of each section appear in a small window. Press Enter or click the contents list to review the entire section.

Once a section has been opened for viewing, it can be browsed line-by-line with the up and down arrow keys, or page-by-page with the PgUp and PgDn keys. The Home and End keys jump to the top or bottom of the section. Press or click Esc to return to the table of contents.

2.6.5 Source Code.

This menu item provides source code listings for the channel and communications modules in COMLNK. A submenu allows the user to choose between assembly language and Fortran implementation of the various code modules. Fortran 77 is the primary development language for COMLNK. For both speed and accuracy, most of the channel and communications modules have been hand-coded in 80x86 assembly language. Only 32-bit code is used in the assembly language modules, which run in protected mode on 386 and higher Intel-compatible processors.

The speed advantage of assembly language is legendary, even in comparison to modern optimizing compilers. What is less well known is that assembly language can also provide an accuracy advantage. In integer operations, for example, assembly language facilitates retaining 64 bits of precision in intermediate results to minimize overflow. In operations involving the floating-point unit (FPU), assembly language facilitates retaining intermediate results in the FPU register stack, where each value retains the full precision of the 80-bit FPU word.

It is recognized that assembly language is something of an anachronism to some programmers and an enigma to most. It is certainly true that development time and portability are not its strong points. For these reasons, all major code modules are first developed in Fortran, and that version is maintained for portability to other platforms. The fact that COMLNK was once ported to the Macintosh is evidence of the success of this approach.[1]

Once the language preference has been indicated, a list of source code modules appears. Use the up and down arrow keys to move through the module list while the first few lines of each module are displayed in a small window. Press Enter or click the module list to look at the entire source code listing.

When a listing has been opened for viewing, it can be browsed line-by-line with the up and down arrow keys, or page-by-page with the PgUp and PgDn keys. The Home and End keys jump to the top or bottom of the listing. Note that there may be more than one routine in a code module. Press or click Esc to return to the list of modules.

[1] The Macintosh is no longer supported because it was found to be too slow for effective application of this program.

SECTION 3
HELP TOPICS

3.1 HOW TO ANALYZE A LAYOUT.

The Analyze layout function in the Run menu checks a link layout on display for proper module placement and consistency of design parameters. Press Ctrl-A to invoke this function from anywhere in the menu system. The function examines the layout and issues a warning if a potential problem is found. If a warning is displayed, the user can decide to correct the problem or ignore the warning.

Note that passing the analysis does not guarantee good link performance. A link designer will most likely need to experiment with a design to produce one that performs well over the range of channel characteristics that pertains to the application. COMLNK does not attempt to automatically find such a design. Rather, it attempts to make the designer's efforts in this regard more efficient.

In general, the user is advised to carefully examine any layout, including its design parameter values, before embarking on an extensive set of simulation runs. It is advisable to first make a few short test runs, watch the error rates on display as the run progresses, and print out the resulting .PRN file. The name of this file is shown at the upper right of the screen during a run.

Examine all inputs and performance results shown in the output. In many cases, examination of the tracking and signal statistics will reveal the cause of poor link performance. It is always a good idea to turn on the plot output file and plot the various values. This will provide considerable insight into modem operation and channel conditions.

Related Topics: Select User interface from the Info menu and review the discussion of the Run menu. See also the help topics on how to plot simulation results and how to print simulation results.

3.2 HOW TO BROWSE FILES.

Any file in any directory on any active drive on the host machine can be browsed, except those that may be currently in use by the operating system or other programs. Files can be browsed while a simulation is running. Any type of file can be browsed, including text and binary files. The display switches automatically between a text format for ASCII files and a hexadecimal format for binary files. One can manually switch between the two formats using the F2 key when the browse function is active.

The browse function is found in the File menu and can be accessed at any time by pressing Ctrl-B. When this function is selected, a list of files that match the file specification appears, if any. The file specification, shown at the lower left of the screen (see Figure 2-1), has the same format as the DOS DIR command. By editing this file specification, any directory on any active drive in the host machine can be accessed.

The file specification may contain the DOS wildcard characters (`*` and `?`). It is set initially to `*.PR?`. This displays a list of all files with extensions that begin with PR in the current directory. This list includes COMLNK print output files.

The file specification can be changed to display other files and to access other directories. For example, a specification of `*.*` displays all files in the current directory, including entries for the parent directory and any existing subdirectories. A specification of `*.` displays only directories and files with no extension.

It is easy to change directories by selecting a directory entry from the list of files, thereby moving up or down the directory tree. For example, selecting the directory entry `<..>` changes to the parent directory, one level closer to the root directory on the current disk drive.

Files can be selected from a different drive by including the drive specification at the beginning of the file specification. For example, a file specification of `A:*.*` displays a list of all files and directories on a floppy disk. Just be sure to insert a properly formatted diskette in the drive before selecting it.

A path can also be added to access a different directory. For example, changing the file specification to `..\*.*` displays a list of all files in the directory preceding the current one. A specification of `\*.*` displays a list of files in the root directory of the current drive. These specifications access other directories without changing the current directory. This is the recommended method of accessing other directories while a simulation is running.

When a list of files is displayed, move the highlight to the name of the desired file or directory and press Enter. Alternatively, type the first few characters of the name until the desired entry is highlighted. Then press Enter to open the file or to change to the directory.

If a mouse is available, a file can be opened for browsing by double-clicking its name in the list, or by highlighting it with a single click and then clicking on the highlighted entry.

If a file is selected, it is opened and displayed on screen. It can then be browsed line-by-line or page-by-page using the cursor movement keys. File browsing does not affect layouts on display and will not impede the progress or otherwise interfere with a simulation that may be running.

A file can be deleted while it is browsed by pressing the Delete key. A prompt will request confirmation of this action. Select OK to delete the file. It will then be "erased" in the usual DOS manner. Note that files deleted under DOS are not moved to the Windows recycle bin.

Related Topics: Select User interface from the Info menu and review the discussion of the File menu.

3.3 HOW TO SPECIFY CHANNEL PARAMETERS.

3.3.1 Mean SNR.

The mean received value of signal-to-noise ratio (SNR) is specified in the modem data. Open the Layout menu and select the Module data function. Ctrl-M provides a shortcut to this function. Move the module highlight to the modulator or demodulator and press Enter to open the modem data menu. The mean SNR is given either in terms of carrier power-to-noise density ratio (C/N_o), or user bit energy-to-noise density ratio (E_{ub}/N_o), at the user's option. Specify the value of either quantity in decibel units. The specified quantity gives the mean level. In a fading channel, the received signal strength fluctuates around this level.

3.3.2 Fading Channel.

Parameters that define the fading channel are specified in the channel data menu. These include the S_4 scintillation index, the scintillation decorrelation time (τ_o), and the frequency selective bandwidth (f_o). To examine or specify the values of these parameters, open the Layout menu and select the Module data function (Ctrl-M). Then move the module highlight to the channel module and press Enter to open the channel data menu.

The S_4 scintillation index specifies the intensity of signal amplitude fading. The value of S_4 may range from zero to unity. An undisturbed nonfading additive white Gaussian noise (AWGN) channel is obtained with $S_4 = 0$. A Rayleigh fading channel is obtained with $S_4 = 1$. Intermediate values, $0 < S_4 < 1$, produce Rician fading, sometimes referred to as "specular plus Rayleigh" fading.

The decorrelation time, τ_o, specifies the channel fading rate or Doppler spread. Small decorrelation times correspond to fast fading with large Doppler spread. Large decorrelation times correspond to slow fading with small Doppler spread. In this context, the terms "fast fading" and "slow fading" are only relevant with respect to the modulation symbol period or other coherent integration time in the receiver. For example, a one-millisecond value of τ_o represents fast fading in a low data rate link, but may be slow fading in a high data rate link.

The value of τ_o should be taken into account when other parameters are specified, especially the dimensions of an interleaver and the length of the simulation run. The interleaver span is often chosen on the basis of the largest expected value of decorrelation time.[1] The run length should encompass several thousand decorrelation times to provide a statistically significant result. Note that the value of τ_o has meaning only in a fading channel; it has no effect when S_4 is zero.

The frequency selective bandwidth, f_o, specifies the channel coherence bandwidth or delay spread. Values of f_o that are small relative to the signal bandwidth correspond to a frequency selective channel with large delay spread. Values of f_o that are large relative to the signal bandwidth correspond to a flat fading channel with negligible delay spread. When frequency hopping is used, values of f_o large compared to the modulation rate but small compared to the hop bandwidth give rise to frequency selectivity over the hop band. In any case, the value of f_o has meaning only in a fading channel; it has no effect when S_4 is zero.

3.3.3 Random Phase.

Other channel parameters enable specification of an additional random phase component, which can represent carrier phase disturbances such as oscillator phase noise or large-scale total electron content (TEC) variations.

The random phase has a Gaussian distribution and is specified by its standard deviation and decorrelation time. Small values of the phase decorrelation time correspond to a relatively wideband random process, while large values yield a slowly varying narrowband process. The phase decorrelation time has no effect when the phase standard deviation is zero.

3.3.4 Channel Realization.

Digital filters are used to generate the random channel impulse response and any additional phase noise that is specified. The filters are initialized in a manner that minimizes start-up transients. However, to ensure statistically stationary results, the filters can be "warmed up" if desired by sampling them prior to the start of a run. A number of warm-up samples corresponding to three or four decorrelation times should be sufficient.

The fidelity of the sampled channel in comparison to a continuous channel is set by the number of samples per decorrelation time (τ_o). We recommend using 20 to 40 channel samples per decorrelation time. Values less than 10 are allowed for comparison with existing hardware channel simulators, but are not recommended.

The specific realization of the fading channel can be changed by specifying the starting random number seed. This enables a run to be repeated with a different channel realization to test the

[1] Guidelines for selection of interleaver dimensions are given in Section 4.14.

statistical significance of measured error rates and tracking loop operation. When a transponder is used, the two channels that are involved should be given different random number seeds to ensure that they are statistically independent.

3.3.5 Range Dynamics.

Range dynamics may result from motion of the transmitter or receiver platform. Three canonical dynamics profiles are available for testing the performance of carrier and time tracking loops. These are sinusoidal, initial-value, and square-wave profiles. With each profile, the user can specify time derivatives of range through fourth order, together with a period (sinusoidal and square-wave profiles) or a duration (initial-value profile).

Range derivatives vary with time in the manner specified and are integrated to form the next lower one. Rather complex profiles can result if several nonzero derivatives are specified. Resulting values are converted to carrier phase and delay dynamics using the carrier frequency specified in the modem data.

Use care in specifying the initial range. Large values can cause immediate loss of delay lock and concomitant link failure if the receiver does not have a time acquisition algorithm.

3.3.6 TEC Dynamics.

TEC dynamics arise from changes in total electron content along a transionospheric propagation path. The same three canonical profiles described above are available to examine tracking loop operation. With each profile, the user can specify TEC derivatives through fourth order along with a period (sinusoidal and square-wave profiles) or a duration (initial-value profile).

TEC derivatives vary with time in the manner specified and are integrated to form the next lower one. Rather complex profiles can result if several nonzero derivatives are specified. Resulting values are converted to carrier phase and delay dynamics using the carrier frequency specified in the modem data.

Use care in specifying the initial TEC. Large values can cause immediate loss of delay lock and concomitant link failure if the receiver does not have a time acquisition algorithm.

<u>Related Topics</u>: Select Code modules from the Info menu and review the material on the channel models. See also the help topics on nonstationary channels, recorded channels, run length, and interleaver dimensions.

3.4 HOW TO SPECIFY A NONSTATIONARY CHANNEL.

Channels can be statistically stationary or nonstationary. Nonstationary channels are encountered when propagation path characteristics vary with time. This can result from changing levels of ionization along transionospheric paths, or from motion of transmitter and receiver terminals.

Either type of channel can be specified in the channel data menu. At the bottom of this menu is a submenu that allows the channel statistics to be toggled from static (stationary) to variable (nonstationary). Recorded channels can also be specified. When nonstationary channel statistics are selected, the channel update interval and the name of the channel parameter data file can be specified.

The channel parameter update interval determines how often the coefficients in the digital filters used to generate the channel impulse response are updated to represent the time-varying channel statistics. File data are interpolated at these update times to provide smoothly varying channel parameters between the data times given in the channel file. Any value can be specified for the update interval. A typical value is one second, which is the program default value.

The channel parameters are given as a function of time in an ASCII data file whose name is specified by the user. The program expects this file to reside in the current directory or in the directory containing the layout data file. If the channel parameter file is not found, a statistically stationary channel with parameter values entered in the channel data menu will be substituted. In this event a warning will appear on the screen, but the run will continue unless terminated by the user.

COMLNK channel parameter files have a modified free-field format. Data records begin with the character string .PATH as an identifier field. Records beginning with any other characters are treated as comments and ignored. Each data record contains up to ten values after the .PATH identifier. The order of the data is significant, but column position is not. If a data record contains fewer than ten items, the values omitted at the end of the record are left unchanged from those entered on a preceding record. An example of a channel parameter file is provided by SCENARIO.CHN on the distribution diskette. The ten items in a channel parameter data record are listed below.

Table 3-1. Channel parameter data record.

Data Offset	Propagation Path Parameter
0	Simulation time (s)
1	Absorption and scattering loss (dB)
2	Excess antenna temperature (deg K)
3	S_4 scintillation index (0 to 1)
4	Antenna-filtered τ_o (s)
5	Antenna-filtered f_o (Hz)
6	Range delta (m)
7	Range rate (m/s)
8	TEC delta (el/m2)
9	TEC rate (el/m2/s)

Note that COMLNK employs the International System of units (SI), previously known as MKS units. This differs from some programs that employ CGS units for quantities such as total electron content. Note also that the values of range and TEC are the changes in these quantities from an initial value. If absolute values of range or TEC are given, the resulting time delay will often cause immediate loss of delay lock in the receiver.

Values of the channel parameters are provided at an arbitrary set of simulation times. An initial simulation time can be specified in the run options submenu of the modem data menu. This initial time enables a run to start at any time relative to the channel data times. Thus a run can start before or after the first channel data time. If the run starts before the first channel time, the static channel parameters given in the channel data menu are used prior to the first channel time.

Related Topics: See the help topics on how to specify modem parameters, how to specify channel parameters, and how to specify recorded channels. Also, select Code modules from the Info menu and review the material on the channel models.

3.5 HOW TO SPECIFY A RECORDED CHANNEL.

Specific realizations of the channel impulse response function may be generated on-line or read in from data files. Use of recorded data allows the channel response to be obtained from external sources. For example, digital samples of the inphase (I) and quadrature (Q) voltages measured in field experiments can be used directly in COMLNK.

A recorded channel can be selected in the channel data menu. At the bottom of this menu is a submenu that allows you to toggle the channel statistics from static to variable to recorded. The first two options employ on-line channel models. The third option specifies the use of recorded channel data. When a recorded channel is selected, the nominal value of the sampling interval and the name of the channel I-Q data file can be specified.

The sampling interval is the time between successive samples of the channel impulse response. This value is normally specified as part of the header record in the channel data file. The value entered via the menu is used only if the sampling interval is not defined in the data file.

The I and Q channel data are given as a function of time in an ASCII file with a user-specified name. The program expects this file to reside in the current directory or in the directory containing the layout data file. If the channel data file is not found, an on-line channel with parameter values entered in the channel data menu will be substituted. In this event a warning will appear on the screen, but the run will continue unless terminated by the user.

COMLNK I-Q channel data files have a modified free-field format. Data order is significant, but column position is not. Each data record may contain up to 32 values, limited to a total of 256 columns.

Valid records start with one of two identifier fields: header records begin with the characters .IQHEAD and data records begin with .IQDATA. In addition, data can be continued from one record to another. Continuation records begin with &IQDATA. Records beginning with any characters other than these three fields are treated as comments and ignored.

The .IQHEAD record specifies sampling parameters for the data that follows. This record contains four values after the identifier field and should precede other data records. It may appear again at any time that the channel sampling parameters change. The four values in a header record are listed below.

Table 3-2. Recorded channel header record.

Data Offset	Channel Sampling Parameter
0	Sampling time interval (s)
1	Number of delay taps (N)
2	Delay tap spacing (s)
3	Hop selectivity flag (0 or 1)

The number of delay taps determines whether the channel response is flat or frequency selective. A single tap provides a flat response, while two or more taps provide a frequency selective response. The maximum number of taps is 64.

The tap spacing is the time delay between successive taps. This value is not currently used because the tap spacing in the COMLNK analog-to-digital converter routines is currently set at one-half the modulation symbol period. Data for frequency selective channels should be sampled at this delay spacing when the hop selectivity flag is set to zero.

The hop selectivity flag is zero when the channel is selective over the modulation symbol period. Setting this flag to one enables the channel to be specified as flat over the modulation symbol period but selective over the frequency-hopping bandwidth. The delay tap spacing is then taken to be the reciprocal of the hop bandwidth.

Subsequent .IQDATA records provide the I and Q samples for each of the N delay taps. Each record, with &IQDATA continuations as needed, provides data for one sampling time. The 2N values in these records are ordered as follows.

Table 3-3. Recorded channel data record.

Data Offset	Channel Response Data
0	I-channel sample, tap 1
1	Q-channel sample, tap 1
2	I-channel sample, tap 2
3	Q-channel sample, tap 2
...	...
2N-2	I-channel sample, tap N
2N-1	Q-channel sample, tap N

The I and Q samples have voltage units. It is recommended that the channel response be normalized to exhibit unit mean power over the entire realization, but this is not a requirement. A normalized channel response makes the mean SNR entered in the modem data menu meaningful. If the response is unnormalized, the mean gain or loss in the channel data will affect the mean SNR. In any case, the mean SNR will be measured when the simulation is run and will be given in the signal statistics section of the print output file.

If fewer than 2N values are given in an .IQDATA record (including continuations, if any), unspecified values at the end of the record are left unchanged from those given previously. Any excess values, more than 2N, are discarded.

If the simulation continues past the time for which the final set of data has been provided, the channel response is repeated from the beginning as many times as required. An example of a sampled channel data file is provided by the file RECORDED.CIQ on the distribution diskette.

Related Topics: See the help topics on how to specify modem parameters and how to specify channel parameters. Also, select Code modules from the Info menu and review the material on the channel models.

3.6 HOW TO SET CODE PARAMETERS.

Two classes of codes are available: convolutional error correction codes, and block error detection codes. A link may incorporate no more than one error detection code. However, any number of error correction codes may be included.

3.6.1 Binary Error Correction Codes.

Binary error correction codes are implemented as constraint length K, rate 1/N convolutional codes. The code constraint length, K, is the number of bits in the encoder shift register and is a measure of code strength. Both the amount of work that the decoder must do and its memory usage increase exponentially with constraint length. The allowable range of K is 3 to 28, but values greater than 18 may exhaust available memory.

A code constraint length K = 7 is a popular choice. We suggest that larger constraint lengths, say K = 8 to 14, also be investigated to determine link performance sensitivity to this design parameter.

The inverse code rate, N, is the number of output code bits produced for each input data bit. This is equal to the number of modulo-two adders in the encoder and is a measure of code redundancy. The channel bit rate increases linearly with the value of N. The allowable range of N is 2 to 30.

The value N = 2 (rate 1/2 code) is a common choice, and is nearly optimum in an undisturbed AWGN channel. However, rate 1/2 codes are not optimum in fading channels (Bogusch, 1989b). Rate 1/3 or 1/4 codes (N = 3 or 4) are much better design choices in fading channel applications. Even lower rate codes (larger N) are desirable to mitigate fast fading effects in low data rate links.

Each of the N modulo-two adders is connected to a specified set of taps on the K-stage encoder shift register. The tap connections are specified by the code generators. These are simply bit patterns where "1" in any bit position denotes a connection and "0" denotes no connection to the corresponding shift register stage. By convention these generators are given as octal numbers.

For given values of K and N, the code generators determine the quality of the code. A set of known good codes is included and can be accessed by entering zero for the first generator. For constraint length 14 or less, this provides one of the best known codes, although there may be others that are just as good. If a constraint length 15 or more is specified, the user should specify his own code generators as the default values are not necessarily very good for $K \geq 15$.

It is possible to specify code generators that yield a catastrophic code. This is one for which the decoder may enter an erroneous state from which it may not emerge. Thus, an unlimited number of decoding errors may be made with a finite number of demodulation errors. A necessary and sufficient condition for a code to be catastrophic is that the code generators have a common factor. A test for this is included and a warning is displayed if a catastrophic code is specified.

Two other parameters are included in the coder/decoder (codec) data menu. These are the maximum quantization level and the path memory length, both of which are design parameters for the maximum-likelihood Viterbi decoder.

The maximum quantization level is the largest symbol metric that can be input to the decoder. Symbol metrics have values from zero to this level. Hence, this quantity specifies hard or soft decision decoding. A maximum quantization level of one yields hard decision decoding, whereas seven provides 3-bit soft decisions. Any value up to 255 (8-bit soft decisions) can be specified. An even value gives an odd number of levels, which yields a central null or "erasure" zone.

While 3-bit quantization (maximum quantization level of seven) is a common choice, better performance may be obtained in fading channel applications using finer quantization. We suggest experimenting with maximum quantization levels of 15 to 255 (4-bit to 8-bit soft decisions).

The decoder path memory bit length specifies the number of retained bits in the path history for each code state. This path length must not be less than the code constraint length. Values around four to five times the constraint length are recommended. However, due to computer word length constraints, the maximum path memory is 32 bits. We suggest that 32-bit paths be used in all cases.

3.6.2 M-ary Error Correction Codes.

When M-ary FSK modulation is used with M equal to four or more, an M-ary code that is precisely matched to the modulation alphabet can be selected. For example, a rate 1/2 code is matched to 4-ary FSK modulation, a rate 1/3 code is matched to 8-ary FSK, and a rate 1/4 code is matched to 16-ary FSK. In general, a rate 1/N code is matched to M-ary FSK modulation when $N = \log_2(M)$. When this condition is satisfied, the code bits for each data bit form the M-ary symbol. This enables the M matched filter outputs from the demodulator to be used directly as the soft decision symbol metrics in the Viterbi decoder.

Therefore, COMLNK provides the option of using M-ary codes matched to M-ary FSK modulation. The number of modulo-two adders, which is equal to the number of code bits produced for each input data bit, must be equal to the number of bits in the M-ary FSK symbol. The allowable range of values is two to seven, which matches 4-ary to 128-ary FSK modulation.

3.6.3 Error Detection Codes.

Two types of error detection block codes are available: cyclic redundancy check (CRC) codes and Hamming codes. CRC codes are generally more robust and are recommended for most applications. Hamming codes are still found in some systems and are included as an option. Each link in a layout may incorporate either a CRC code or a Hamming code, but not both.

3.6.3.1 CRC Codes. A CRC code involves specification of the number of information or data bits in each code block and the number of parity or check bits added at the end of each block. Any number of information bits per block can be specified. The number of parity bits per block must be in the range from one to 32.

A CRC encoder, if present, must be the first transmit module after the source module. The specified user bit rate is the rate at which the parity-encoded block is transmitted. The source data bit rate is reduced by the ratio D/(D+P), where D is the number of data bits per CRC block and P is the number of parity bits per block.

Parity check bits are generated by passing data bits through a linear feedback shift register (LFSR). The feedback connections are specified by a generator polynomial. This is simply a bit pattern where "1" in any bit position denotes a feedback connection and "0" denotes no connection to the corresponding shift register stage. The CRC generator is displayed in hexadecimal format and may have any number of bits up to the number of parity bits.

The initial contents of the CRC shift register are specified by the LFSR preset. This is a bit pattern that is loaded into the shift register prior to generating parity for each block. It is displayed in hexadecimal format and may have any number of bits up to the number of parity bits. Zero may be used, but a nonzero value is recommended to detect erroneous all zero blocks.

3.6.3.2 Hamming Codes. A Hamming code involves specification of the number of information or data bits in each code block and the number of parity or check bits added at the end of each block. Here, the number of parity bits is constrained to the range from four to six, and the number of data bits must be in the range from five to 26 and also depends on the number of parity bits.

A Hamming encoder, if present, must be the first transmit module after the source module. The specified user bit rate is the rate at which parity-encoded blocks are transmitted. The source data bit rate is reduced by the ratio D1/(D2+K), where D1 is the number of data bits in a short block, D2 is the number of data bits in a long block, and K is the number of parity bits per block.

The use of two block sizes arises from the need to accommodate specialized variations of the Hamming code, such as that used to encode the navigation data in the Global Positioning System (GPS). The user can select one of five built-in options. These are the GPS NAV (32,26) specialized code, conventional (15,11) and (31,26) Hamming codes, and common (16,11) and (32,26) extended Hamming codes. Other custom variations can be constructed, as discussed in Section 4.8.

Related Topics: Select Code modules from the Info menu and review the material on binary and M-ary convolutional codes, CRC codes, and Hamming codes.

3.7 HOW TO EDIT A LAYOUT.

The layout from any file can be edited, or a completely new layout can be constructed using any combination of available modules. Files can be opened and read in and layouts can be edited and saved while a run is in progress.

To edit a layout on display, open the File menu and select Edit this layout (or press Ctrl-E). If the displayed layout is not running, this is equivalent to opening the Layout menu directly. The Edit function enables a running layout to be edited and can be used in all cases.

To edit a background layout not on display, open the Display menu and select swap background (Ctrl-W) to get it on display. Then press Ctrl-E or open the Layout menu.

To edit a layout previously stored in a file, open the File menu and select Open layout file (Ctrl-O). Then select the desired file. Once the desired layout is on display, press Ctrl-E or open the Layout menu.

To build a new layout, open the Layout menu and select the Copy/cut/paste menu item. Then select Delete this layout. If this function is unavailable because the layout is running, first select swap background and delete the background layout, if any.

Once the Layout menu is open, select Insert to build a new layout or to add one or more modules in an existing layout. Select Delete to delete modules from the layout. Select Replace to change a module from one type to another. Select Module data to specify or examine the design data for each module. Help is available for each function; press F1 at any time to display help messages.

Modules from a layout on display can be copied or cut using functions available in the Copy/cut/paste submenu. These modules can then be pasted into the same layout or into a different layout. This makes it easy to duplicate parts of a layout or to work on two layouts simultaneously, one in the foreground and the other in the background.

Using the menu functions is by far the easiest way to edit and save layout data files. However, these files can also be edited off-line using any text editor that can read and write plain ASCII files. Comments are included in the files COMLNK.DAT and DEMO.DAT to assist those who wish to edit layouts off-line.

Related Topics: Select User interface from the Info menu and review the discussion of the Layout menu.

3.8 HOW TO TERMINATE A SIMULATION RUN.

A run will terminate automatically once it attains the specified number of user bits or user bit errors, whichever is reached first. These values are part of the source module data. If a running layout contains more than one link, each link will terminate automatically when it attains the number of user bits or bit errors specified in the source module for that link.

To terminate a run prematurely, select the Terminate link function in the Run menu. Ctrl-T provides a shortcut to this function. All links in the running layout will terminate immediately once the decision is confirmed. The results of the run will be written to the print file. If the layout file contains a series of cases, the next case will start automatically. Otherwise, the program will await further instructions.

If a running layout contains more than one link, each link can be individually terminated by entering the link number using the Alt-keypad procedure. For example, if only the second link in a layout is to be terminated, hold down the Alt key and press and release 2 on the numeric keypad; then release the Alt key. Link 2 will then terminate, leaving the other links in the layout running. Note that this only works with the numeric keypad, not with the top row of number keys. Also, the keyboard NumLock light must be on.

When all links in a layout are finished, the next case will start automatically if the program is running from an input file that contains a series of cases. If the user does not want to complete the series, each case can be terminated after it starts using the Terminate link function. Alternatively, the entire series can be terminated and the display cleared using the Program reset function in the Run menu. The layout on display is automatically saved to disk if it has been modified, and therefore can be read back in.

Related Topics: Select User interface from the Info menu and review the discussion of the Run menu. Also see the help topic on how to quit the program.

3.9 HOW TO GET HELP.

Context-sensitive help is always available by pressing the F1 key, or by clicking the right mouse button. Frequent use of this feature is highly recommended as the user gains familiarity with program operation and capabilities.

The F1 key or right mouse button is a toggle; it first displays a help message and then removes it. Once help is invoked by pressing F1 or right-clicking the mouse, help messages continue to be displayed while menu selections are changed. Pressing F1 or right-clicking a second time cancels the help display. Help messages are also canceled by pressing Esc or starting a run.

Help may be enabled automatically as the user navigates through the menus, depending on the amount of experience. The COMLNK help system keeps track of usage and turns help on frequently for new users. This automatic help feature gradually phases out as the user gains familiarity with the program. Help can be invoked manually at any time by pressing F1 or clicking the right mouse button.

Related Topics: Additional information is provided by the on-line documentation accessible from the Info menu and contained in this document. Additional information and specialized assistance may be available from the program developer or sponsoring agency. The sponsor and developer can be contacted at the following addresses.

Sponsor:	***Developer:***
Defense Threat Reduction Agency	Mission Research Corporation
6801 Telegraph Road	P. O. Box 2256
Alexandria, VA 22310-3398	Atascadero, CA 93423-2256
Attn: Dr. Kenneth Schwartz/TDANP	Attn: Mr. Robert L. Bogusch
Voice: (703) 325-1096	Voice: (805) 461-0235
Email: kenneth.schwartz@dtra.mil	Email: rlbogusch@earthlink.net

3.10 HOW TO DISPLAY RESULTS GRAPHICALLY.

Graphical displays of simulation results can be obtained by post-processing a plot data file that is optionally written during a simulation run. The file, which has the same name as the print file but with an extension of .BIN, is a binary file that contains performance data as a function of time during the run. Quantities such as received signal characteristics, jamming interference, tracking loop operation, and data demodulation can be plotted. The quantities written to the plot file can be selected from an extensive plot data menu.

To obtain a plot output file, open the Layout menu and select Module data (Ctrl-M provides a shortcut). Move the module highlight to the modulator or demodulator and press Enter to open the modem data menu. Then open the run options submenu at the bottom of the modem menu. Note the plot on and off times. If the plot off time is later than the on time, a plot file will be written if at least some of the simulation occurs in the specified interval.

Once a plot file is enabled, a plot data selection menu can be opened at the bottom of the modem run options submenu. A large number of quantities can be selected from this menu for plot file output. Choose as few or as many as desired by moving the cursor and toggling the selection with the Enter key or the spacebar. A default set of quantities is also available. Press F1 for additional information.

If the layout contains more than one modem, plot files can be turned on for any or all of them. The plot times and plot quantities can be selected individually for each modem in the layout. Follow the foregoing procedure for each modem. Separate plot files will be written for each of the selected modems.

The binary file format is intended to be used by an on-line graphics package that is not yet installed in the program. This file is not directly readable by most graphics programs and therefore must be converted to another format. A utility program, BIN2PLT.EXE, is included on the distribution diskettes to convert the binary plot file to an ASCII format that can be imported into a spreadsheet or other plot program.

Related Topics: See the help topic on how to plot simulation results.

3.11 HOW TO SET INTERLEAVER PARAMETERS.

Interleaving is a form of time diversity, used to break up demodulation error bursts before they are fed into an error correction decoder. Interleaving is effective only when an error correction code is incorporated in the link.

In the transmit chain, an interleaver reorders the sequence of encoded symbols. In the receive chain, a complementary deinterleaver restores the original order prior to decoding, thereby shuffling the order of demodulation errors. Two types of interleavers are in common use: block interleavers and convolutional interleavers.

3.11.1 Block Interleaver.

A block interleaver is a two-dimensional (row-column) array. The interleaver writes into columns and reads out of rows. Hence, the number of interleaver columns is related to channel memory (decorrelation time), and the number of interleaver rows is related to code memory (constraint length and code rate). Any number of columns up to 1,048,575 and any number of rows up to 2,048 can be specified, subject to available memory as discussed below.

The interleaver span is defined to be the number of rows times the number of columns, divided by the rate at which symbols pass through the interleaver. For good performance in fading channels, the interleaver span should exceed 30 times the largest value of scintillation decorrelation time

(τ_o) that will likely be encountered. A reasonable choice for the number of rows is around two to three times the product of the code constraint length, K, and the inverse code rate, N. The number of columns is then set to yield the desired span.

Interleaving can be done using a fixed sequence or in a pseudorandom manner. Pseudorandom interleaving involves scrambling the interleaver row read order and the deinterleaver row write order. When attention is focused on fading channel performance, fixed interleaving is generally preferable if the interleaver span is much larger than the largest value of τ_o with the number of rows and columns chosen in accordance with the above guidelines. Pseudorandom interleaving may be advantageous if the interleaver dimensions are suboptimal, or to mitigate the effects of jamming interference.

For synchronous operation, two arrays are used in the transmitter and receiver. Hence, block interleaving has the disadvantage of requiring twice the delay and about four times the memory of an equivalent convolutional interleaver. On the other hand, block interleaving offers the advantage of block repetition, wherein each interleaved block is read out two or more times. Recombining the repeated symbols in the deinterleaver provides the potential for diversity gain in fading channels.

Block repetition is specified by the block repeat factor, which is the number of times that the interleaver array is read out and transmitted prior to filling it with the next set of encoded symbols. This method of repetition provides maximum time separation between repeated symbols, and can provide a significant advantage over conventional symbol repetition in slow fading channels.

It is possible to specify interleaver dimensions that cause interleaver size to exceed the amount of memory available to the COMLNK dynamic storage system. As each module is linked into the layout at run time, the pool of available storage is reduced by the memory requirements of that module. If there is sufficient memory for the module, it is initialized and the process continues. If not, the run stops with an error message on the screen. In that case, the size of the interleaver must be reduced by lowering the number of rows and/or columns. Alternatively, the user may reduce the overall complexity of the link layout in some other manner. If neither of these options is acceptable, then it will be necessary to contact the developer to obtain a recompiled version of COMLNK that allocates a larger amount of memory to dynamic storage.

3.11.2 Convolutional Interleaver.

A convolutional interleaver can be described as a two-dimensional (row-column) array that has been sliced diagonally, with one half placed in the transmitter and the other half in the receiver. A single array gives synchronous operation with one-half the delay and about one-fourth the memory of a block interleaver (FSK modems using soft-decision decoding require additional deinterleaver memory in both designs).

The number of columns is the longest row length, which is related to channel memory (decorrelation time), while the number of rows is related to code memory (constraint length and code rate). The span is again defined as the number of rows times the number of columns, divided by the rate at which symbols pass through the interleaver. The guidelines given above for choosing the block interleaver dimensions apply here as well.

Related Topics: See the help topics on how to set code parameters, how to set channel parameters and how to get help.

3.12 HOW TO SPECIFY JAMMER/RFI PARAMETERS.

3.12.1 Waveform.

Jammers or other sources of radio frequency interference may radiate continuous or pulsed broadband or bandlimited noise, an array of CW or pulsed tones, or linear-FM (chirp) pulses. To install a jammer, open the Layout menu and select the Insert module function (Ctrl-I provides a shortcut to this function). Move the insert cursor to the demodulator input or to the end of the channel (the End key facilitates this). Move the cursor in the module selection list to Jammer. Then press Enter to install the jammer in the layout. It is possible to install more than one jammer at a demodulator if the RFI environment is best represented by the superposition of different waveforms.

Once a jammer is installed, open the Layout menu and select the Module data function (Ctrl-M provides a shortcut to this function). Move the module highlight to the jammer and press Enter to open the jammer data menu. Press F1 to activate context-sensitive help if it is not already on display.

A switch at the top of the jammer data menu cycles through the basic waveform options: noise, CW tone, pulsed noise, pulsed tone, and chirp pulse. Depending on this selection, additional parameters are available to specify the jammer frequency band, pulse timing, and other quantities described below.

3.12.2 Jammer Power.

For all waveforms, the mean received jammer power is specified as a ratio to the mean received signal power, in decibels. This specified value of J/S defines the total power in the jammer frequency band. For pulse jammers, the average (not the peak) power should be entered.

3.12.3 Frequency Band.

The specification of the jammer frequency band depends on the waveform. For a noise jammer, the center of the jammer frequency band can be offset from the center of the system frequency band if the system is frequency hopped. Then, full-band or partial-band jamming is specified by the ratio of jammer bandwidth to hopping bandwidth. The noise jamming power is spread evenly over the resulting frequency band. If the system does not hop, the noise jammer power is spread over the signal modulation bandwidth or A/D sampling bandwidth, whichever is larger.

For a tone jammer, the jammer frequency band is defined by the spacing between a specified number of equally spaced tones. The resulting frequency band can be displaced from the system center carrier frequency by a specified offset. If an odd number of tones is specified, the center tone is displaced from the center carrier by the given offset. With an even number of tones, the two center tones are displaced above and below this offset frequency by half the tone spacing.

A tone jammer can have any number of tones with any frequency spacing. Jamming power is allocated uniformly over the specified set of tones. Exercise care in specifying the number of tones and spacing; tones outside the system bandwidth simply reduce jammer effectiveness.

Multiple jammer tones can be coherently related at the transmitter, or they can be noncoherent with a random initial phase distribution. This jammer design option can have a remarkable effect on the nature of the resulting interference waveform when large numbers of tones are radiated.

For a chirp pulse, the jammer frequency band is defined by the pulse compression ratio together with the center frequency offset. The pulse compression ratio is the time-bandwidth product BT of the linear-FM pulse. In other words, the ratio is equal to the swept bandwidth in Hertz times the pulse duration in seconds. A positive value signifies up-chirp (increasing frequency sweep), while a negative value signifies down-chirp (decreasing frequency sweep).

3.12.4 Pulse Jammer.

A pulse jammer is characterized by the pulse repetition frequency (PRF) and the pulse duty cycle. The duty cycle is the ratio of pulsewidth to pulse repetition interval (reciprocal of the PRF). Thus the pulsewidth is equal to the duty cycle divided by the PRF.

Pulse jammers may be frequency followers, if the communication system uses frequency hopping. Between pulses, a follower jammer listens to the transmitted signal and emits the next pulse at or near the observed system frequency. The time delay between the signal and the jammer is determined from a range offset given in the dynamics submenu (discussed below). If the jammer is located too far from the path between the transmitter and receiver, the frequency follower capability will not be effective.

Because a frequency follower observes the transmitted frequency, which with FSK modulation contains the data modulation offset, the jammer frequency offset can be used to avoid placing the jammer energy precisely on top of a modulated tone.

3.12.5 Jammer Dynamics.

Jammer Doppler and delay dynamics can result from platform motion. Three canonical range dynamics profiles are available: sinusoidal, initial-value, and square-wave profiles. For each profile the range offset, velocity, and acceleration can be specified, along with a period for the sinusoidal or square-wave profiles or a duration for the initial-value profile.

The specified velocity and/or acceleration vary with time according to the selected profile and are integrated to yield a range offset and velocity time history. The resulting range and velocity determine the jammer time delay and Doppler shift as a function of time.

The jammer range offset takes on special significance when frequency follower capability is specified for the jammer, as noted above. The range offset determines the time delay between the signal path and the jammer path. If this delay exceeds the signal hop duration, frequency following is ineffective.

3.12.6 Jammer Scintillation.

The propagation path between jammer and receiver can be nonfading or fading, independent of the signal path. The S_4 scintillation index determines the intensity of jammer amplitude fading. No fading along the jammer path results with $S_4 = 0$. Rayleigh fading results with $S_4 = 1$. Intermediate values, $0 < S_4 < 1$, produce Rician fading, which may range from weak (S_4 near zero) to strong (S_4 near unity).

The decorrelation time, τ_o, specifies the jammer channel fading rate. Small decorrelation times correspond to fast fading while large values correspond to slow fading. The value has no effect when $S_4 = 0$.

3.12.7 Jammer Antenna.

With any waveform and any propagation path, the jammer antenna can be fixed or rotating. A rotating antenna produces periodic interference at the receiver. This effect can be produced by a friendly or hostile radar, for example.

When a rotating jammer antenna is specified, the antenna beamwidth becomes an important parameter. The effective beamwidth is the angular separation between the half-power points on the jammer radiation pattern, as measured at the communications receiver.

A *sin(x)/x* gain pattern is built in and is normally used to characterize the jammer antenna pattern. This can be changed to a user-specified antenna gain pattern, provided in the form of a table. A gain table cannot be entered via the interactive user interface. It must be provided by off-line editing of the layout data file. Comments in the COMLNK.DAT and DEMO.DAT files provide additional information for users who wish to specify a jammer antenna gain table. When a gain table is provided, it will be used if a zero value is entered for the beamwidth. This is the only situation in which a zero beamwidth is allowed with a rotating antenna.

Related Topics: Select Code modules from the Info menu and review the material on the jammer models.

3.13 HOW TO SET RUN LENGTH.

The run length is set by two parameters in the source/sink module data menu. These are the maximum number of received user bits, and the maximum number of errors in the received user bit stream. The run terminates when either value is reached.

Any number of bits up to 2^{31}-1 can be specified.[1] Any number of errors can also be specified, except that a zero value is replaced by the number of bits. This forces the run to encompass the specified number of user bits regardless of the number of errors.

If a fading channel is specified with stationary statistics (*i. e.*, $S_4 > 0$ with no channel parameter file), the run should encompass several thousand decorrelation times for reasonable statistical significance. For example, if the value of decorrelation time (τ_o) is 1 second, the run length should not be less than several thousand seconds. Thus if the data rate is, say, 1000 bits/s the run should encompass several million user bits or more to achieve a reasonable degree of statistical significance. High data rates (or low error rates) may lead to very long runs in a fading channel.

Another requirement for statistical significance is that there should be at least 100 independent error events. Hence, if errors occur randomly, the run should be long enough for at least 100 user bit errors to occur. However, errors do not occur randomly in a fading channel, nor in any link in which error correction coding is employed. Even with random errors at the decoder input, a decoder tends to make errors in short bursts. Consequently, the number of independent error events is considerably less than the total number of errors. Therefore, as a general rule, it is recommended that a run encompass at least 1000 user bit errors for decent error-rate statistics.

Even longer runs may be required if the desired measure of performance is a message or block error rate, rather than the bit error rate. Again, there should be at least 100 independent error events to achieve reasonable statistical significance, but the requirement now is defined in terms of message or block errors.

One way to evaluate whether the error-rate measurements in a run have converged to a statistically significant result is to plot the cumulative average error rates as a function of time. Such a plot enables the number of independent error events to be estimated and reveals the effect of each event on the cumulative error rate.

[1] All counters use 32-bit signed words, for which the largest positive value is $2^{31} - 1 = 2{,}147{,}483{,}647$. Because the channel bit rate exceeds the user bit rate in almost all cases, the channel bit counter and other intermediate counters will likely saturate when the maximum number of user bits is specified. No counters will overflow, and all counts will be correct up to and including the point of saturation.

Related Topics: See the help topics on how to set channel parameters, how to plot simulation results, and how to estimate run times.

3.14 HOW TO SET MODEM PARAMETERS.

There are by far the largest number of design parameters available in the modem data menu. Not only are there a selection of FSK and PSK waveforms, each modem also involves up to four tracking loops (more for PN/PSK modems when started in an acquisition mode). Each tracking loop has a number of user-specifiable design parameters. Link performance is often quite sensitive to tracking loop design. The A/D converter design values can also strongly affect demodulator and decoder performance.

Information on most of the modem parameters is available in the on-line help, accessed by pressing F1 when a parameter in the data menu is highlighted. The sample data files distributed with the program can be used as templates in initial selection of modem parameter values. These values should be reexamined if the data rate is changed or whenever the link design is modified.

Several of the modem parameters have predefined sets of allowable values. These choices are displayed by pressing the Enter key, which cycles through the values that can be selected. Examples include the choice of absolute or differential phase encoding and binary or quaternary modulation in a PSK modem.

3.14.1 Signal Level.

The mean received signal strength can be specified in terms of the ratio of carrier power to one-sided noise spectral density (C/N_0), or in terms of the ratio of user bit energy to noise spectral density (E_{ub}/N_0). Use whichever quantity is more convenient. In either case, the value is entered in decibel units (dB-Hz or dB, respectively). Note that in a fading channel the actual received signal level will fluctuate above and below the specified mean level.

The design-point signal level is also specified in the modem data menu. This is the value of C/N_0 at which the quantizers and tracking loops in the receiver are designed to operate. Do not confuse this parameter with the received signal strength just discussed. The design-point C/N_0 is a design parameter and should be left unchanged once the proper value is set. Two options are available in cases where the design-point C/N_0 is not known: (1) enter zero to allow the program to use a default value, or (2) experiment until a value is found for which the receiver exhibits satisfactory performance over the expected range of received signal levels. In any case, if a signal-based AGC loop is operative, it will adjust the receiver gain in an attempt to hold the actual signal level near the specified design level at the tracking loop inputs.

3.14.2 AGC Parameters.

Several AGC options are available, including the option to disable the AGC loop. In any case, the design-point gain can be set. This is the value of an 8-bit gain control word corresponding to the design-point signal level just discussed. If the AGC is turned on, it will stabilize at this gain if the actual received signal strength is equal to the design level.

Because an 8-bit gain control word can take on values from one to 255, the design point can be specified anywhere in this range. The gain control word is a voltage multiplier, and thus the total dynamic range of the AGC is $20\log_{10}(255) = 48$ dB. The design-point gain allocates this dynamic range between gain reduction and gain increase. A design value of 128, for example, limits the possible increase in gain to a maximum of 6 dB, while allowing the gain to decrease by as much as 42 dB. A design value of 255 causes the AGC to operate as a variable attenuator, with no ability to respond to a decrease in input signal level.

The AGC design-point noise level is the one-sigma value of the digitized noise in the I and Q channels of the A/D converter when the received signal-to-noise ratio is equal to the design level and the AGC gain is at its design value. Inasmuch as signal and noise voltages are just numbers in the A/D output, it is convenient to refer to their units as quanta. The design level of noise quanta can be specified in the range from one to 255. Together with the specified design value of C/N_0 and the specified design-point gain, this number sets the design value of signal-plus-noise quanta in the A/D output.

A large design-point value of noise quanta enables weak signals to be resolved, but increases the possibility of quantizer saturation with strong signals. A small value of noise quanta reduces the chance of saturation, but increases the possibility of quantizer bottoming with weak signals. If the proper value of this parameter is not known, either experiment to find a value that gives good performance over the expected range of received signal levels, or let the program select a default value.

Other AGC and quantizer parameters tend to be specialized for either FSK or PSK waveforms. For example, the clipping level and power scale factor can be set with FSK modulation, while the quantizer scale factor and nonlinearity can be set with PSK modulation. Use the F1 help key to obtain information on these and other parameters.

Much more information on the general topics of modulation, demodulation, and tracking is contained in the references listed at the end of the main body of this report (Bogusch, 1989b; Bogusch, 1990).

Related Topics: Select User interface from the Info menu and review the discussion of the Layout menu. Also select Code modules from the Info menu and review the technical discussions of the A/D converter, AGC, DLL, FLL, PLL, and the specific modulator and demodulator modules of interest.

3.15 HOW TO MODIFY LAYOUTS AND MODULES.

A layout can be modified by editing an input file or by constructing a new layout from scratch. Layout files can be opened and read in and layouts can be edited and saved while a simulation is running.

The design parameters associated with each module can be varied over rather wide ranges to effectively modify the module. For example, the convolutional encoder module can be made to provide any rate 1/N code with constraint length K, where code rates from 1/2 through 1/30 and constraint lengths from 3 through 28 can be specified. For each combination of code rate and constraint length, any set of code generators can be specified. The decoder module automatically reconfigures itself to provide maximum-likelihood Viterbi decoding of the specified code.

Therefore, as long as the available modules with their parameter ranges encompass the desired design, modification is simply a matter of input selection.

The situation is more complicated if a module not currently included in COMLNK is needed. For example, while Reed-Solomon codes are planned for a future release, they are not currently implemented in COMLNK. If such a code is required or some other capability not currently available is needed, the user should contact the developer or sponsor and request a code modification to include the desired capability. Modifications and additions that enhance the utility of the program to a broad spectrum of users are most eagerly sought.

Related Topics: Select User interface from the Info menu and review the discussion of the Layout menu for information on editing layouts. Also see the topic on how to get help for information on how to contact the code developer and sponsor.

3.16 HOW TO SET UP MULTIPLE LINKS.

To set up a layout with multiple independent links, open the Layout menu and select the Insert module function. Insertion of a source module starts a new link that is completely independent of the other links in the layout. There is no limit to the number of links that may be simultaneously simulated, other than the amount of available memory.

Once the source/sink module pair has been inserted for a new link, simply insert the other modules that comprise the link, together with their design parameters. When the layout is executed , each link will run for a length of time determined by the run length settings in its source data menu. Input data and results for each link are summarized in the print output file.

An easy way to set up multiple links that have many features in common is to use the copy and paste functions in the Layout menu Copy/cut/paste submenu. Set up the first link and use the copy function to copy the entire link. Then use the paste function to paste in each of the copied modules. This will produce two identical links. Now use the other Layout menu functions to make the changes necessary in either link to complete the layout.

The sample data file TWOLINKS.DAT on the distribution diskette provides an example of a layout containing two independent links. This layout can be edited to form the desired configuration, and additional links can be inserted. Indeed, any layout can be edited to incorporate additional links simply by inserting or copying modules as described above.

Related Topics: Select User interface from the Info menu and review the discussion of the Layout menu. Also, see the help topics on editing layouts, setting the run length, terminating runs, and printing results.

3.17 HOW TO OPEN LAYOUT DATA FILES.

Any layout data file in any directory on the host machine can be opened and read in . Files can be opened and read in while a simulation is running without affecting the run. Thus the next series of cases can be set up while the progress of the current simulation is observed.

The file open function is found in the File menu and can be accessed at any time by pressing Ctrl-O. When this function is selected, a list of files that match the file specification appears, if any. The file specification, shown at the bottom of the screen (see Figure 2-1), has the same format as the DOS DIR command. By editing this file specification, any directory on any active drive in the machine can be accessed.

The file specification may contain the DOS wildcard characters (`*` and `?`). It is set initially to `*.DA?`. This displays a list of all files in the directory with an extension beginning with DA, which includes the example data files distributed with the program. There is no requirement to use the .DAT extension for layout data files. Any file naming convention can be used.

The file specification can be changed to display other files and to access other directories. For example, a specification of `*.*` displays all files in the current directory, including directory entries. A specification of `*.` displays only directories and files with no extension.

The current directory is easily changed by selecting a directory entry from the list of files, thereby moving up or down the directory tree. For example, selecting the directory entry `<..>` changes to the parent directory, one level closer to the root directory on the current disk drive.

Files on a different drive can be selected by including the drive specification at the beginning of the file specification. For example, the file specification `A:*.*` displays all files and directories on a floppy disk. Be sure to insert a properly formatted diskette in the drive before selecting it.

A path can also be added to the file specification to access a different directory. For example, changing the file specification to `..\*.*` lists all files in the directory preceding the current one. A specification of `\*.*` lists files in the root directory of the current drive. These specifications access other directories without changing the current directory. This is the recommended method of accessing layout files in other directories when a simulation is running.

When a list of files is displayed, move the highlight to the name of the file or directory that you want and press Enter. Alternatively, type the first few characters of the name so that the desired entry is highlighted. Then press Enter to open the file or change to the directory.

If a mouse is available, a file can be opened by double-clicking its name in the list. Or, the filename can be highlighted with a single click. Pressing Enter or clicking on the highlighted entry then opens the file.

Once a file is selected, it is opened and read in. If the file is actually a COMLNK layout data file, the first layout that it contains will immediately be displayed on the screen. If the file is not a data file, a message to that effect will be displayed, whereupon a different file can be selected.

If there already was a layout on display, it is not lost when a new layout is read in. The previous layout is simply moved to the background. If there already was a background layout, it is not lost; it is written to a file if not saved previously. The user never needs to worry about losing layout data no matter how many files are opened and read in.

Related Topics: Select User interface from the Info menu and review the discussion of the File menu. Also see the help topic on how to read in a layout.

3.18 HOW TO PLOT SIMULATION RESULTS.

Simulation results can be plotted by post-processing a plot data file that can be written during a simulation run. The file, which has the same name as the print file except that the extension is .BIN, is a binary file that contains performance data as a function of time. The data written to the plot file are selected from an extensive menu list and may include received signal and jammer characteristics, tracking loop operation, and data demodulation performance.

To cause a plot file to be written, open the Layout menu and select Module data (or press Ctrl-M). Open the module data menu for the modulator or demodulator and then open the run options submenu. Note the plot on and off times. If the plot off time is later than the on time, a plot file will be written if at least some of the simulation occurs in the specified plot interval.

The frequency at which data are written to the file is controlled by specifying the number of A/D samples between successive plot outputs. The default setting, obtained by entering zero, produces approximately 2048 time points during the specified plot interval.

Once a plot file is enabled, the plot data selection menu at the bottom of the modem run options submenu can be opened. A large number of quantities can be selected from the menu list for plot output. A default selection appears the first time that the plot menu is opened, and the selected quantities are marked in the list. The selection can be changed by moving the highlight to any item and pressing the Enter key or the spacebar to select or deselect it. As few or as many quantities as desired can be selected. The Insert key selects all items in the list, and the Delete key clears the current selections and restores the original defaults.

If the layout contains more than one modem, such as in a concatenated link, separate plot files with individually selected data can be produced for each modem. Plot files can be turned on and off individually for each modem by opening each run options submenu and appropriately setting the plot on and off times. The time intervals and output data can be specified differently for each modem by following the procedure outlined above.

Table 3-4 lists the quantities that appear on the plot data selection menu list. Any combination of these quantities can be written to the plot data file as a function of time during the specified plot interval. The simulation time in seconds is always the first quantity written at each time and therefore does not appear on the menu list.

Table 3-4. Plot data selection menu list.

Signal Amplitude (dB)	AGC Gain (dB)
Signal Delay (symbols or PN chips)	Delay Error (symbols or PN chips)
Signal Delay Rate (Hz)	Delay Rate Error (Hz)
Signal Doppler (Hz)	Frequency Error (Hz)
Signal Phase (radians)	Phase Error (radians)
Delay Rate Estimate (Hz)	Doppler Estimate (Hz)
Delay Estimate (symbols or PN chips)	Hop Frequency Offset (Hz)
Transmit Modulation	Demod Hard Decision
Modulation Phase (cycles)	Phase Error (cycles)
I-Channel Sample	Q-Channel Sample
FLL Lock Detector Output	PLL Lock Detector Output
DLL Lock Detector Output	OQPSK/MFSK Lock Detector Output
Demod BER (cumulative)	Demod BER (running average)
Decoder Input BER (cumulative)	Decoder Input BER (running average)
Decoder Output BER (cumulative)	Decoder Output BER (running average)
User BER (cumulative)	User BER (running average)
Character Error Rate (cumulative)	Character Error Rate (running average)
Block Error Rate (cumulative)	Block Error Rate (running average)
Packet 1 Error Rate (cumulative)	Packet 1 Error Rate (running average)
Packet 2 Error Rate (cumulative)	Packet 2 Error Rate (running average)
Packet 3 Error Rate (cumulative)	Packet 3 Error Rate (running average)
Packet 4 Error Rate (cumulative)	Packet 4 Error Rate (running average)
Packet 5 Error Rate (cumulative)	Packet 5 Error Rate (running average)
Jammer Power, J/S (dB)	Jammer Doppler (Hz)
Jammer Phase (cycles)	Jammer Delay Rate (Hz)
Jammer Pulse Time (samples)	Jammer Rotation (degrees)
Jammer Range (meters)	Jammer Velocity (meters/second)
Jammer Channel Amplitude (dB)	Jammer Channel Doppler (Hz)

The jammer parameters at the bottom of this list appear only if the modem is affected by jamming interference. Any or all of the packet sizes may be set to zero in the source/sink data menu; the error rate items for zero-length packets are grayed out in the plot data selection menu.

Some of the items in this list require frequent outputs to the plot data file in order to produce useful plots. For example, it is generally advisable to write out every A/D sample when transmit modulation, demodulator hard decisions, modulation phase, or I and Q channel A/D samples are to be plotted. This can produce very large files unless the plot data interval is quite short. Consequently, it is advisable to give careful consideration to the specification of plot interval, output frequency, and data items for each application.

The format of the binary plot data files is designed for use by on-line graphics routines that are not yet integrated in COMLNK. The binary files cannot be read by most graphics programs and must be converted to another format. A file conversion utility program, BIN2PLT.EXE, is included on the COMLNK distribution diskettes. This utility can be run from the DOS command line to convert binary plot data files to an ASCII format that can be imported into a spreadsheet or other graphics package.

Related Topics: See the help topic on how to display results graphically.

3.19 HOW TO PRINT SIMULATION RESULTS.

Every simulation run is documented in a print output file with a .PRN extension. COMLNK automatically assigns output filenames so that they have the same name as the layout data file, with the extension changed to .PRN. If the layout was constructed on-line and not yet given a name by saving it, the print output file is assigned the name LAYOUT.PRN. In any case, the name of the output file is displayed at the upper right of the screen during the run.

If an output file with the same name existed previously, the previous file is saved by changing the last character in the extension. This procedure ensures that files are not overwritten, but are saved with extensions that reflect the order in which they were created. The oldest file in the series will have an extension .PR0, the next oldest will have an extension .PR1 and so on, with the most recent print file having the .PRN extension.

Files can be renamed as desired using the DOS or Windows rename commands. Unneeded files can also be deleted using DOS or Windows delete commands or by means of the delete option available in the Browse function of the File menu. It is recommended that output files produced during production runs be archived and saved.

The print file summarizes all of the input quantities that define the link layout, along with the final results. Results include cumulative error counts and error rates, tracking error statistics, signal statistics, and jammer statistics. The file is a standard ASCII text file that can be browsed on the monitor screen or sent to a printer. Assuming that a printer is connected to the host machine, a simple way of printing this file is to issue the following command at the DOS prompt:

```
COPY filename.ext PRN
```

where filename.ext is the name and extension of the specific file to be printed. Alternatively, the Notepad utility included with Windows can be used to browse and print COMLNK print files.

The COMLNK information files, having an extension .INF, can also be printed if desired. These are basically ASCII text files with a special header record. A printed copy of these information files can be obtained by first loading them into a word processor such as the Windows WordPad utility. Then simply delete the header record and send the file to the printer. To ensure that special mathematical and graphical characters are properly printed, the IBM PC extended ASCII symbol set should be selected prior to printing (consult the printer manual). Be sure to make a backup copy of the original information file first; this file must not be changed.

Related Topics: Select User interface from the Info menu and refer to the discussion of the Info menu system information files. Also, see the help topics on how to find statistical summaries and how to print the user manual.

Example Of Simulation Results.

Excerpts from the print output file produced by running the example data file JAM16FSK.DAT are included here to illustrate the type of information provided by each simulation run.

The first section of the print output is a formatted list of all of the inputs, including link design values, channel parameters, and jammer parameters. The name of the input layout data file is shown at the top along with the date of the run, the COMLNK version number, the layout identification label, the time of the run, and the name of the print output file.

The inputs for each module in the layout are then summarized in a two-column format. The modules are listed in the order that they appear in the transmit chain. Data that the program derived from the inputs, including default values that are computed at the start of the run, are included in the tabulation of input values.

For example, see the section below that summarizes the modulator/demodulator parameters. The A/D sampling rate and channel bit and symbol rates are shown, along with the corresponding signal-to-noise ratios. The following notation is used for the signal-to-noise ratios:

E_{ds}/N_o:	*energy-to-noise density ratio for digital A/D samples*
E_{cb}/N_o:	*energy-to-noise density ratio for channel bit period*
E_{cs}/N_o:	*energy-to-noise density ratio for channel symbol period*
E_{ub}/N_o:	*energy-to-noise density ratio for user bit period*
E_{fh}/N_o:	*energy-to-noise density ratio for frequency-hop period*

For some modems, the symbol E_{hop}/N_o is used for the hop energy-to-noise density ratio. For a binary modem, the channel bit rate and symbol rate are the same, but are different for an M-ary modem as shown here. The relationships between user bit rate, channel bit rate, channel symbol rate, and frequency-hop rate depend on the details of the coding, multiplexing, and modulation techniques employed in the system. In all cases, the mean signal-to-noise ratios for all rates are always summarized in the print file.

Units are indicated for all quantities where appropriate. In general, the program uses the International System (SI), formerly known as the MKS system of units. While the communications modules in COMLNK operate internally in pure integer mode, many inputs are provided as real numbers. The real numbers are shown in the printout; conversion to integer format with appropriate scaling is accomplished internally by the module initialization routines.

In this context, attention is directed to the section below that lists the tracking loop design parameters (AGC, DLL, FLL). The real numbers that are provided as input are listed, together with the scaled integer coefficients that the program actually uses to implement the tracking loops. A positive binary shift count corresponds to a left shift, while a negative shift count indicates a right shift.

If the layout contains more than one link segment, such as an uplink-downlink configuration, each segment is identified and the modules in that segment are listed with their inputs. For multiple-link layouts, each link is identified and the modules in that link are listed in the order of appearance in the layout. The following is a relatively simple single-segment single-link layout.

```
------------------------------------------------------------------------

JAM16FSK.DAT  Sat 12 Feb 2000   COMLNK 5.3    JAM_16-FSK 09:12:22 JAM16FSK.PRN

---------------------- Link Number 1 ----- Segment Number 1 ------------------

                                SOURCE/SINK
User Data Bit Rate (Hz)  1.92000E+04      Fixed Source File Name  JAM16FSK.XMT
Maximum No. Bits/Trial        100000      Number of Source Chars            72
Maximum No. Bit Errors        100000      Bits per Source Char               6
Number of Bits/Block             360      Screen Text Option       Errors Only
Number of Bits/Packet 1          512      Output Text Option      Display None

First Line of ASCII Text:
16-ARY FH/FSK IN RAYLEIGH FADING WITH PULSED PARTIAL-BAND TONE JAMMING..

                            CRC ENCODER/DECODER
Number CRC Parity Bits            32      Total Bits/CRC Block             360
Data Bits in CRC Block           328      CRC Polynomial (hex)        04C11DB7
                                          LFSR Preset    (hex)        FFFFFFFF

                 BINARY/M-ARY CONVOLUTIONAL ENCODER/VITERBI DECODER
Code Constraint Length             8      Path Memory Bit Length            32
Number of Mod-2 Adders             4      No. of Decoder Metrics            16
Octal Connect Patterns     231   277   335   353

                    CONVOLUTIONAL INTERLEAVER/DEINTERLEAVER
Interleaver Type        Fixed Symbol      Interleaver Span (syms)        20000
Number of IL Rows                 40      Interleaver Span (s)     1.04167E+00
Number of IL Cols                500      Interleaver Delay (s)    1.04375E+00
No. De-IL Quantities              16      Interleaver Span/Tau0    1.04167E+01

                          FSK MODULATOR/DEMODULATOR
Type of Modulation        16-ary FSK      Type of Demodulation     DFT Filters
Center Frequency (Hz)    1.00000E+09      C/No Mean Value (dB-Hz)  5.48330E+01
A/D  Sample Rate (Hz)    6.14400E+05      Eds/No Mean Value (dB)  -3.05150E+00
Channel Bit Rate (Hz)    7.68000E+04      Ecb/No Mean Value (dB)   5.97940E+00
Channel Sym Rate (Hz)    1.92000E+04      Ecs/No Mean Value (dB)   1.20000E+01
User    Bit Rate (Hz)    1.92000E+04      Eub/No Mean Value (dB)   1.20000E+01
FSK Tone Spacing (Hz)    1.92000E+04      Tone Spacing/Sym Rate              1
Random Noise Seed (hex)     D19C6200      A/D Rate/Symbol Rate              32

Frequency Hop Rate (Hz)  1.92000E+04      Efh/No Mean Value (dB)   1.20000E+01
Hopping Bandwidth  (Hz)  1.96608E+07      Number of Symbols/Hop              1
User Channel Allocation  1.00000E+00      User Channel Rate (Hz)   1.92000E+04

AGC Type:                      Noise      FSK Detector: Normalized       Power
Design C/No (dB-Hz)      6.08948E+01      Design I,Q Gain Factor            64
Design Signal Quanta              90      Design Noise Quanta               45
Actual Signal Quanta              45      Filter Clipping Level            255
AGC Time Constant (s)    1.00000E+00      Filter Scale Factor                8
AGC Update Rate  (Hz)    3.61412E+04      Number Samples/Update             17
                                          Rcv Noise Temp (deg K)   2.90000E+02
AGC Shift Count                   23      AGC Control Value               7594

DLL Type:    Precise Early-Late Data      Clk NCO Resolution (Hz)  2.28882E-03
DLL Order                          2      Data Symbols per 1000           1000
DLL Update Rate  (Hz)    1.55340E+00      Number Symbols/Update          12360
DLL Bandwidth (Hz)       1.00000E-01      DLL Damping Factor       7.07107E-01
Coef 1 Shift Count               -10      Coef 1 Multiplier                 16
Coef 2 Shift Count                -7      Coef 2 Multiplier                 21
Accum1 Shift Count                -7

FLL Type:    Precise OffsetTone Data      Car NCO Resolution (Hz)  9.15527E-03
FLL Order                          2      Data Symbols per 1000           1000
FLL Update Rate  (Hz)    3.20000E+03      Number Symbols/Update              6
FLL Offset Tone (Hz)     9.60000E+03
FLL Bandwidth (Hz)       1.00000E+00      FLL Damping Factor       7.07107E-01
Coef 1 Shift Count                -4      Coef 1 Multiplier                 18
Coef 2 Shift Count                -7      Coef 2 Multiplier                 15
Accum1 Shift Count                -4      Accum2 Shift Count                -8
```

```
                               FADING CHANNEL
Type of Channel         Flat Rayleigh       S4 Scintillation Index     1.00000E+00
Type of Fading Spectrum           f-4       Decorrelation Time (s)     1.00000E-01
Freq Selective BW (Hz)    1.00000E+09       Signal Chip Period (s)     5.20833E-05
Hopping Bandwidth (Hz)    1.96608E+07       Number of Delay Taps                 1
Channel Update Rt (Hz)    4.00000E+02       Number of Samples/Tau0              40
Number Channel Warmups              0       Random Number Seed (hex)      F4D1A937

                                 JAMMER/RFI
Jammer Waveform            Pulsed Tone      Frequency Offset (Hz)      1.50000E+06
Number Random i Tones              307      Tone Spacing     (Hz)      1.00000E+04
Jammer Power J/S (dB)      5.00000E+01      Min. Jammer Freq (Hz)      9.99970E+08
Jammer Bandwidth (Hz)      3.06000E+06      Max. Jammer Freq (Hz)      1.00303E+09
System Bandwidth (Hz)      1.96608E+07      Bandwidth Ratio            1.55640E-01
Equiv. Ecb/(N+J) (dB)     -2.59204E+01      Random No. Seed (hex)         F261E3EC

Pulse Jammer PRF (Hz)      1.00000E+01      Pulse Spacing      (s)     1.00000E-01
Pulse Duty Cycle           1.00000E-01      Pulse Duration     (s)     1.00000E-02
Pulse Initial Delay        8.56581E-01      Pulse Delay Time   (s)     8.56581E-02
Jammer Rotation (rpm)      7.80000E+01      Jammer 3dB Beam (deg)      1.00000E+01

Jammer Path S4             2.70000E-01      Decorrelation Time (s)     1.00000E+00

Total Modules in Layout             18      Top of Dynamic Memory           851788
Number of Comm. Modules             12      First Unused Location            46949
Number of Other Modules              6      Number of Spare Words           804839
```

The jammer parameters shown above appear whenever a jammer is included in the layout. Several of these values are derived from user inputs and are included in the print file for information purposes. For example, the system bandwidth is the total bandwidth utilized by the communications system. For a frequency-hopped system it is the hopping bandwidth. In direct-sequence spread-spectrum systems it is the PN code rate. If spread-spectrum modulation is not used, it is the larger of the modulation bandwidth or the A/D sampling rate.

The entry just below the system bandwidth is the noise equivalent value of the (channel bit energy)/(noise density plus jammer energy) ratio. This value is computed as if the jammer radiated white Gaussian noise with uniform density over the system bandwidth. This equivalent value of E_{cb}/N_o is not used in the simulation. It is included in the print file because it provides a figure of merit to assist in evaluating the effectiveness of the actual jammer waveform.

The six values in the last group of data shown above provide information on the number of modules and amount of memory required to run the simulation. The total number of modules includes the communications, channel, and jammer modules whose inputs are listed above, together with other modules installed to measure error rates and accumulate signal and tracking error statistics. These statistical outputs appear farther down toward the end of the print file.

The next section in the output provides information on the precise layout of each module used to execute the simulation. The code name of the module is given, together with the interrupt clock timing and I/O bus pointers.

The first number under the module name is the delay, measured in interrupt clock cycles, prior to the first interrupt for that module. The second number is the number of interrupt clock cycles between subsequent interrupts for that module. The interrupt clock frequency is shown below at the right.

The last four numbers under the module name are the I/O bus offsets for the input and output dynamic datasets, the link number associated with the module, and the interrupt rate. The latter value is equal to the clock frequency divided by the number of clock cycles between interrupts.

```
                                    LINK LAYOUT         Clock (Hz)      2.45760E+06
MODUL      SYSOPS     SOURCE     CRCCOD     NCODEM     ILCNVI     MFSKTX     CHANL1
KLOCK           0          1          2          3          4          5       2053
KDELT         128        128        128        128        128        128       6144
IOPT1           0        148       1146       1232       1241       1251        368
IOPT2          12       1146       1232       1241       1251       3956       4096
LNKNO           0          1          1          1          1          1          1
R(Hz)     19200.0    19200.0    19200.0    19200.0    19200.0    19200.0      400.0

MODUL      CHANL1     JAMMER      MFSK2      STATS     ERRFSK     ILCNVD     DCODEM
KLOCK        2053       2054       2055       2056       2180       2180    2567301
KDELT       61440          4          4       1280        128        128        128
IOPT1        1087       1077       6742       6209       6209       6209       6816
IOPT2        4121       4146       6209       6747       6800       6816      46241
LNKNO           1          1          1          1          1          1          1
R(Hz)        40.0   614400.0   614400.0     1920.0    19200.0    19200.0    19200.0

MODUL      ERRMCS     ERRDCB     CRCCHK       SINK
KLOCK     2567301    2571270    2571270    2571271
KDELT         128        128        128        128
IOPT1        6816      46241      46241      46864
IOPT2       46782      46829      46864      46873
LNKNO           1          1          1          1
R(Hz)     19200.0    19200.0    19200.0    19200.0
```

The next section in the print output file provides the summary of measured error rates for each link. Error rates are measured at the output of each demodulator in the layout, at the inputs and outputs of most error-correction decoders, and at the user data sink. Measurements at the data sink include the user bit error rate, character or word error rate (depending on ASCII or binary source format), CRC or user block error rate (depending on whether a CRC encoder is present), and packet error rates for up to five packet sizes specified as part of the source module data.

Each error rate in the print file is a cumulative measure, equal to the total number of errors divided by the total number of items received. These total counts are also shown. Error-rate summaries are provided for each link in a multiple-link layout. Running averages for all of the measured error rates in each link are available in optional plot data output files.

```
LINK 1 SUMMARY:          M-ary FSK Demod    Encoded    Decoded       User      6-Bit
                       Symbols       Bits       Bits       Bits       Bits      Chars
  Number Received       129799     519196     439036     109728     100000      16666
  Number in Error        26229      55810      47718         88         66         40
  Mean Error Rate    2.021E-01  1.075E-01  1.087E-01  8.020E-04  6.600E-04  2.400E-03

                       360-Bit    512-Bit
                     CRC Block    Packets
  Number Received          304        195
  Number in Error           34         27
  Mean Error Rate    1.118E-01  1.385E-01
```

The final section in the output file, presented below, provides the summaries of measured tracking error statistics and measured signal statistics for each link segment.

Tracking error statistics include the mean, root-mean-square (rms), standard deviation, minimum value, and maximum value of the error in each quantity for which a tracking loop is, or could be, included in the demodulator. These statistics are accumulated for signal time delay error, carrier frequency error, carrier phase error, and AGC operation.

A similar statistical summary is provided for the received signal. Signal statistics are accumulated for received signal time delay, carrier frequency, carrier phase, and signal-to-noise ratio (user bit energy-to-noise density ratio). The measured value of the mean-square log-amplitude is included for the signal under the heading Chi^2.

If a jammer is present, statistics are also accumulated for the actual measured value of jammer interference in the A/D samples. This J/S measurement is made in the A/D sampling bandwidth with the sampled jammer waveform, and therefore includes the effects of jammer scintillation, antenna rotation, pulse timing, and frequency offset.

These statistics are accumulated by a module that is installed at the output of each demodulator in the layout. The number of times that the statistics module sampled the received signal and demodulator outputs is shown at the right side. Time histories of the tracking errors, received signal, and jammer interference are available in optional plot data output files.

```
---------------------- Link Number 1 ----- Segment Number 1 ------------------

CHANNL  Starting Seed       F4D1A937      Ending Seed (hex)              B48A067D
JAMMER  Starting Seed       F261E3EC      Ending Seed (hex)              79C9FCC4
MFSK2   Starting Seed       D19C6200      Ending Seed (hex)              06A7EAE2

FSK Tracking Stats      Mean        RMS       Sigma     Minimum    Maximum     Samples
  Delay (symbols)    1.307E-02 1.415E-02 5.426E-03 0.000E+00 2.063E-02       12980
  Frequency  (Hz)   -2.101E+03 2.371E+03 1.099E+03-3.578E+03 4.330E+00
  Phase (radians)   -3.572E+04 4.587E+04 2.878E+04-8.923E+04 5.334E+00
  AGC Change (dB)   -3.034E+00-2.251E+00           -1.612E+01 9.016E-01

FSK Signal Stats        Mean        RMS     Sigma/S4   Minimum    Maximum     Samples
  Delay (symbols)    0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00       12980
  Frequency  (Hz)    1.186E-01 7.770E+00 7.769E+00-1.951E+02 1.793E+02
  Phase (radians)   -2.204E-02 3.620E+00 3.620E+00-6.280E+00 6.283E+00     Chi^2
  Eb/No       (dB)   1.206E+01           9.122E-01-2.460E+01 1.896E+01 4.455E-01
  Jammer J/S (dB)    2.042E+01           4.751E+01-3.000E+02 5.634E+01

CPU Time (s)         1.532E+01                     Simulation Time (s)       5.208
CPU Rate (user b/s)  6.526E+03                     CPU/Simulation Time       2.942
```

The last four values shown above provide the simulation run timing data. The actual computer (CPU) time required by the run is given, together with the effective run speed in user bits per second. The simulation time and ratio of CPU time to simulation time are also given. For reference, this output file was produced by a 400-MHz Pentium II machine.

3.20 HOW TO QUIT COMLNK.

The user can quit this program and return to the operating system at any time by opening the File menu and selecting Quit program. Ctrl-Q provides a quick way of quitting from any place in the menu system. A prompt will request confirmation of the action to prevent accidental exit.

When running under the DOS prompt, the program exits to the command line. When running in a DOS box under Windows, it may be necessary to close the COMLNK window after quitting.

If the program is quit while a simulation is running, the simulation is terminated and a print file is produced documenting the results as of termination. The name of the print file is shown at the upper right of the screen. All open files are properly closed.

If a layout was modified but not saved, COMLNK automatically saves it under the name AUTOSAVE.DAT before quitting. If there is already a file with the name AUTOSAVE.DAT, the existing file is renamed with a slightly modified extension. Thus, no data is lost when the program is quit.

Related Topics: Select User interface from the Info menu and review the discussion of the File menu. Also, see the help topic on how to print simulation results.

3.21 HOW TO READ IN A LAYOUT.

A layout data file can be read in from any directory. Files can be opened and read in while a simulation is running without disturbing the run. Thus, the next series of cases can be set up while the current run continues.

A layout on display is not lost when a new layout is read in. It is either moved to the background if the background layout is not running, or it is saved to a file. If there is already a background layout that is not running but has been modified, it is saved to a file to allow the current foreground layout to be moved to the background when a new layout is read in.

By this process, the user can read in as many layouts as desired with no danger of losing data. At any point after two layouts have been read in, one will be in the foreground and another in the background. If no layout is running, these will be the two most recently read in. If a layout is running, it may be moved to the background but it is never removed. The foreground and background layouts can be exchanged at any time using the swap background function in the Display menu (press Ctrl-W).

The open function is found in the File menu and can be accessed at any time by pressing Ctrl-O. When this function is selected, a list of files that match the file specification appears, if any. The file specification, shown at the lower left of the screen (Figure 2-1), has the same format as the DOS DIR command. By editing this file specification, any directory on any active drive in the host machine can be accessed.

When a list of data files is displayed, move the highlight to the name of the desired file. Alternatively, type the first few characters of the name to highlight the desired file. When the Enter key is pressed, the highlighted file is opened and the first layout that it contains is displayed. If the file is not a layout data file, a message to that effect will be displayed, whereupon another file can be selected.

Related Topics: Select User interface from the Info menu and review the discussion of the File menu. Also, see the help topic on how to open layout data files.

3.22 HOW TO SET UP CONCATENATED RELAY LINKS.

To build a layout with partially or fully processed concatenated link segments, such as a satellite relay link, open the Layout menu and select Insert module. Position the insertion point highlight in the receive chain of the first link segment at the point at which the receive data is to turned around and processed through the transmit chain of the second link segment. Insert the first module of this second transmit segment. Continue inserting the other transmit modules in the second segment, together with their design parameter values. Do not insert a source module except to start a new, independent link.

One can continue to add link segments in this manner to construct quite complex end-to-end relay links. There is no limit on the number of link segments that may be concatenated, other than that imposed by available memory. The files UPDOWN.DAT, UPXDOWN.DAT, and 2RELAYS.DAT on the distribution diskettes provide a few examples of concatenated link layouts.

An easy way to build a concatenated link whose uplink and downlink functions are similar involves the use of the Copy/cut/paste functions in the Layout menu. First read in or construct a single link having the desired transmit and receive modules. Use the Module data function to enter data for each module. Then open the Copy/cut/paste submenu and use the copy function to copy all of the modules that are to be used in both the uplink and downlink. Press Esc when the copy is complete to return to the submenu. Then open the paste function. A list of the copied modules will appear at the right side of the screen, and an insertion point highlight will be displayed. Position the insertion highlight in the receive chain at the output of the uplink receive module that will be connected to the first downlink transmit module. Then simply paste in each of the copied modules. This will produce a concatenated layout whose uplink and downlink modules are identical. Then modify the modules as needed using the Module data function to complete the layout. This technique can be practiced using the COMLNK.DAT file.

To build a bent-pipe transponder relay link (one that only frequency translates and retransmits the received data), insert a transponder module after the first channel module. Then insert a second channel module between the bent-pipe transmitter and the downlink receiver. The file BENTPIPE.DAT on the distribution diskettes provides an example of such a layout.

Related Topics: Select User interface from the Info menu and review the discussion of the Layout menu. Also, see the help topic on editing layouts.

3.23 HOW TO RUN A SIMULATION.

To initiate simulation of a link layout on display, select the execute layout function in the Run menu. Ctrl-X provides a shortcut to this function. The execute layout function automatically performs the Analyze layout function before launching the run.

The run begins immediately if no errors are detected in analysis of the layout. A detected error causes a warning to be issued. The user then has an opportunity to go to the Layout menu to correct the error, or he may choose to ignore the warning and proceed with the run.

Warnings that may have been ignored in a previous analysis of the layout will be displayed again before the run begins. This provides another opportunity to either correct a problem or ignore the warning. Note that some errors are so severe that, if the user attempts to ignore them, the warning will be displayed again and the run will not commence.

Related Topics: Select User interface from the Info menu and review the discussion of the Run menu. Also, see the help topics on how to read in a layout, how to analyze a layout, how to set the run length, how to plot simulation results, and how to print simulation results.

3.24 HOW TO ESTIMATE RUN TIMES.

An estimate of the time remaining in a currently running layout can be obtained by opening the Display menu and selecting the function How much longer. Ctrl-H provides a convenient shortcut to this function.

This function displays the current elapsed run time, estimated time to complete the run, effective run speed in user bits per second, and the current percent completion. The accuracy of the time estimate improves as the run progresses. Once this estimate is on display, it can be updated by pressing any key, such as the spacebar. The Esc key closes the display. The function can be reselected any time while the run is in progress.

Alternatively, the speed of a run can be estimated by manually timing the user bit rate. Observe the user bit counter on the screen and measure the time required to process some number of bits. In general, the longer the interval that is timed, the greater the accuracy of the measurement. Once the rate at which the run is progressing has been estimated, divide this effective user bit rate into the number of user bits specified in the source module data menu.

If a similar layout has been run previously, it can be used to estimate the running time by examining the time that it required. Open the File menu and use the Browse function (Ctrl-B) to look at the print output file from the previous run. The CPU time for each run is shown at the end of the print file.

Related Topics: Select User interface from the Info menu and review the discussion of the Display menu. Also, see the help topics on how to browse files, how to set the run length, how to terminate a run, and how to print simulation results.

3.25 HOW TO SAVE A LAYOUT.

To save the layout on display, open the File menu and select Save this layout. Ctrl-S provides an easy way of saving the layout from any point in the menu system. This function provides the functionality of both the "save" and "save as" functions found in other programs.

When the Save function is selected, a filename for the layout is suggested. This will be the name of the original input file, if any, or the name under which the layout was saved most recently. If this name is correct, select OK from the dialog box. A confirmation that the layout has been saved will appear on screen.

If it is desired to save the layout under a different name, select Rename to edit the filename. Layout files may have any name acceptable to DOS with any extension, not just .DAT. Different extensions can be used to organize data among several projects, for example. Once the filename is the one desired, select OK. A confirmation will again appear on the screen.

Do not worry about overwriting a file; COMLNK automatically backs up an existing file having the same name by renaming the existing file with a slightly modified extension. Even previous backups are saved in this manner. No data is lost by saving a layout, no matter how many times the same name is used.

If a layout has been changed but not saved when the program is quit, it is automatically saved under the name AUTOSAVE.DAT.

Related Topics: Select User interface from the Info menu and review the discussion of the File menu.

3.26 HOW TO SET UP A SERIES OF RUNS.

There are two ways to set up a series of runs. One way is to set up a main layout file that contains data for the first case in a series. This file then transfers control to a script file that contains just the data that changes for subsequent cases. The files SCRIPT.DAT and SCRIPT.RUN on the distribution diskettes illustrate this method. The main layout file can be created and edited using the Layout menu. At the present time, the script file must be created and edited using an off-line ASCII text editor. There are plans to improve the scripting process in a future release. In the meantime, several examples of script files, with an extension of .RUN, are included with the program and can be edited as needed. Note that the extensions of the layout and script files can be changed to anything that one wants.

The second way to create a series of runs does not involve an off-line editor. Use the Layout menu to prepare the file for the first case in a series. While in the Layout menu select Module data, open the data menu for any modem in the layout, then open the run options submenu at the bottom of the modem data menu. Enter the name of a file that will contain the data for the second case and toggle the script file status to active. When the layout is finished and ready for the first case, save the file under some name. Then make any changes necessary for the second case, including the name of a file that will contain the third case, then save it under the name to which the first case refers. Proceed to make the changes for the third case, including the name of a file that will contain the next case, and save it under the name to which the preceding case refers. In this way an unlimited number of cases can be linked together, each in its own file.

Using either method, the main layout file must first be saved to disk and then read back in to activate the script. To run the series of cases, go to the File menu and open the file that contains the first layout in the series. Then go to the Run menu and execute the layout. The first case will start running. When it finishes, it will automatically initiate the second case. In similar fashion, as each case ends the next case in the series starts automatically. When the last case ends, a message to this effect will appear on screen.

Related Topics: Select User interface from the Info menu and review the discussion of the Layout menu. Also see the help topics on how to edit a layout and how to save a layout.

3.27 HOW TO SELECT ERROR RATES FOR DISPLAY.

When a simulation is running, a set of measured error rates is displayed at the bottom of the screen. The frequency at which this error-rate display is updated is controlled by the number of bits between updates. This quantity is specified prior to the start of the run in the source/sink data menu, display options submenu.

The size of the screen display allows up to six error rates to be shown at any time. The program automatically selects an initial set for display. In a large complex layout there may be many more than six measurements that could be displayed. To select the ones to be updated on the screen while a run is in progress, open the Display menu and select the function pick error rates. Ctrl-K provides a convenient shortcut to this function.

The pick error rates function is only available while a simulation is running. When this function is selected, a list of all of the error-rate measurements being accumulated appears. Those currently being displayed are marked in the list. Any or all of these can be deselected by moving the highlight to a marked measurement and then pressing Enter or the spacebar. The deselected error rate will instantly disappear from the screen.

Once fewer than six error rates are on display, others can be selected by moving the highlight to a desired measurement and pressing Enter or the spacebar to mark it for display. The selected error rate will instantly appear on screen and proceed to be updated at the frequency specified in the source data menu.

The error-rate display can also be scrolled through the list of measurements using the left and right arrow keys. The entire set of measurements can be scrolled off the screen with the right arrow key. The left arrow key then restores the display. Again, the measurements being displayed at any time are marked in the list and can be individually selected and deselected as described above.

Regardless of how many or which of the error rates are displayed, all of the measurements continue to be accumulated during the run. At the end of the run, all error rates are written to the print file. Changing the error-rate display has absolutely no effect on the measurements.

Related Topics: Select User interface from the Info menu and review the discussion of the Display menu. Also, see the help topic on how to print simulation results.

3.28 HOW TO FIND STATISTICAL SUMMARIES.

Every simulation run produces a print output file. The print filename is derived from the layout filename by changing the extension to .PRN. Layouts that produce multiple print files, such as the example file DEMO.DAT, form a series of .PRN files derived from the layout filename. The names of the layout and print files are displayed at the upper left and upper right of the screen, respectively, while the run is in progress. If a file with the same name as the print file already exists, the existing file is saved by renaming it with a different last character in the extension.

A number of statistical summaries are contained in the print file. These include cumulative error rates, carrier tracking statistics, and received signal and jammer statistics. Running average error rates along with time histories of tracking errors and signal characteristics are available in optional plot data output files.

Cumulative error rates summarized in the print file include demodulation bit error rates, bit error rates at the input to an error correction decoder, decoded bit error rates, user bit error rates, CRC block error rates, and packet error rates. For each of these, the total number received and the total number of errors are recorded. The measured cumulative error rate is simply the ratio of these two numbers.

Tracking error statistical summaries include delay error, frequency error, phase error, and AGC operation. The mean error, rms error, minimum error, and maximum error are summarized for each tracking loop.

Received signal statistical summaries include signal delay, Doppler shift, phase shift, and signal amplitude. Again, the mean value, rms value, minimum value, and maximum value of each quantity are summarized. The measured value of the mean-square log-amplitude is also provided for the signal.

If a jammer is present, a statistical summary of the measured value of jammer interference is included in the print file. Interference is measured in the A/D sampling bandwidth with the sampled jammer waveform. The measurement includes the effects of jammer scintillation, antenna rotation, pulse timing, and frequency offset.

Information on simulation run time is also provided at the end of the print file. The CPU time and effective user bit rate are given there, along with the simulation time and the ratio of CPU time to simulation time.

Related Topics: See the help topics on how to print simulation results and how to plot simulation results.

3.29 HOW TO SWAP BACKGROUND AND FOREGROUND LAYOUTS.

COMLNK enables the user to work on two layouts simultaneously. One layout is displayed on the monitor screen and is referred to as the foreground layout. The other layout, referred to as the background layout, is not displayed but is retained in memory. If a second layout has not been read in or created on-line, the background layout will be blank.

The foreground and background layouts can be interchanged at any time, regardless of whether either is currently running. To do this, open the Display menu and select swap background. An easy way to do this from anywhere in the menu structure is to press Ctrl-W.

The layout identification bar at the top of the screen, just below the menu bar, shows the name of the file, if any, associated with the displayed layout. The layout bar also displays the layout identification label. If the layout is running or has been run, the layout bar also shows the date and time at which the run commenced. The name of the print output file produced by the run is also shown. This information enables the user to keep track of which of the two layouts is currently on display.

For convenience in working with two layouts, the swap function is also available in View mode and in the Layout menu. When editing layouts in the Layout menu, foreground and background layouts can be swapped by opening the Copy/cut/paste submenu.

Related Topics: Select User interface from the Info menu and review the discussions of the Display menu and the Layout menu. Also see the help topic on how to view a layout.

3.30 HOW TO PRINT USER MANUAL.

An on-line user manual is provided by the Info menu, which displays the information contained in this document. The information provided by this menu is stored in a set of files distributed with the program. These files have names in the format COMLNK#x.INF, where x is a letter denoting the specific type of information in the file. The information files are basically standard ASCII text files with a special header record that the program uses to access and display the file contents.

A printed copy of the information files can be obtained using a word processor. First make a copy of these files (the original information files must not be altered). Load the files into the word processor and delete the header record at the top of each file. At this point the files can be formatted if desired. When they are ready to be printed, select the IBM PC extended ASCII symbol set on the printer (see the printer manual for instructions on selecting the symbol set). This is necessary for proper printing of mathematical and drawing characters in the files. Then send the file to the printer.

The information files can be printed in any order. It is suggested that the printed copies be assembled in the following order to construct the user manual.

COMLNK#I.INF	--	Introduction
COMLNK#U.INF	--	User Interface
COMLNK#H.INF	--	Help Topics
COMLNK#P.INF	--	Signal Propagation Channel Technical Reference
COMLNK#M.INF	--	Digital Communications Modules Technical Reference
COMLNK#F.INF	--	Fortran Source Code
COMLNK#A.INF	--	Assembly Source Code

The number of pages in this user manual will depend on the font size used, as well as on any formatting that may be employed. When printed in a compressed font with minimum formatting, around 100 to 150 pages of text are produced by the first five files listed above. The Fortran source code requires over 300 pages, and the assembly listings require more than 250 pages to print. It is recommended that a fixed-pitch font be used to print code listings, with tabs set at 8-column intervals starting with column 9.

Related Topics: Select User interface from the Info menu and refer to the discussion of the Info menu system information files. Also, see the help topic on how to print simulation results.

3.31 HOW TO VIEW A LARGE LINK LAYOUT.

A complex layout can easily exceed the size of the display window. However, the layout can be scrolled around on the screen to view all parts of it. To do this using the keyboard, open the Display menu and select the function View this layout. Ctrl-V provides a shortcut to this function from anywhere in the menu structure.

When view mode is active, the arrow keys scroll the layout horizontally and vertically on the screen. The Home and End keys facilitate moving to the extremes of the layout. The PgUp and PgDn keys can be used to swap the foreground and background layouts. The Esc key exits view mode.

When a mouse is available, it is not necessary to enter view mode in order to scroll a layout. The layout can be scrolled using scroll buttons at the right side and bottom of the display. Clicking these buttons moves the layout on the screen exactly like the arrow keys do in view mode.

Layouts can be scrolled at any time, regardless of whether a simulation is running. Scrolling a layout does not affect the operation or speed of a run in progress.

Another way of viewing a layout is to select the Module data function in the Layout menu. Ctrl-M provides a shortcut to this function. Once the Module data function is selected, the arrow keys can move the module highlight to each module in the layout.

Related Topics: Select User interface from the Info menu and review the discussions of the Display menu and the Layout menu.

3.32 HOW TO RUN IN A DOS WINDOW.

When running under Windows 95 or 98, COMLNK can be run either in full screen mode or in a DOS window. If the user chooses to run in windowed mode, either by invoking COMLNK at the DOS prompt or by double-clicking the COMLNK icon that may have been installed on the Windows desktop, it may be necessary to adjust the properties of that window to obtain the desired behavior.

To set the window properties, right-click on the title bar at the top of the DOS window and then click on Properties at the bottom of the pop-up menu. In the Program tab, put a check mark in the box labeled Close on exit. Click Apply.

Next, click the Misc tab and look in the Background section. Make sure that the box labeled Always suspend is NOT checked. If there is a check mark in this box, clear it by clicking on the box. Click Apply and then click OK to close the menu.

There are a number of other parameters that can be set in the Properties menu of the DOS window. In our experience, the default settings of these other parameters has proven to be satisfactory. A user who wants to review or change the properties of the DOS prompt should consult the Windows documentation.

Related Topics: See the help topic on how to install COMLNK on the Windows desktop.

3.33 HOW TO INSTALL COMLNK ON WINDOWS DESKTOP.

When running under Windows 95/98, COMLNK can be installed on the desktop for convenient access. This can be accomplished as follows.

Step 1. Install COMLNK on the hard disk.

If COMLNK has already been installed on the hard disk, proceed to Step 2. Otherwise, follow the procedure outlined in the quick-start manual (README.TXT – reproduced in the Appendix). The installation procedure simply involves copying the contents of the distribution disks to a directory on the hard disk. This can be done at the DOS prompt or using Windows. For example, if the hard disk is drive C: and if it is desired to install COMLNK in directory C:\COMLNK, the following sequence of commands can be entered at the DOS prompt:

```
C:                                [log on to hard drive]
CD\                               [change to root directory]
MD COMLNK                         [make COMLNK directory]
CD COMLNK                         [change to COMLNK directory]
(insert disk 1 into drive A)
COPY A:*.*                        [copy all files from disk 1]
(remove disk 1, insert disk 2)
COPY A:*.*                        [copy all files from disk 2]
(remove disk 2, insert disk 3)
COPY A:*.*                        [copy all files from disk 3]
```

Change A: to B: in these commands if the 3.5-inch floppy drive is drive B. If COMLNK is installed in a directory other than C:\COMLNK, change references to this directory in Step 2 below to reflect the drive and directory actually used.

Step 2. Create shortcut on Windows desktop.

In Windows 95/98, double-click My Computer on desktop.
Double-click drive C.
Double-click COMLNK folder.
Right-click COMLNK.EXE file.
Click Create Shortcut in pop-up menu.
Drag shortcut to COMLNK onto desktop.
Close My Computer and any other windows opened in this step.

Step 3. Name the shortcut.

Right-click shortcut to COMLNK.
Click Rename in pop-up menu.
Type COMLNK.
Press Enter.

Step 4. Set shortcut properties.

Right-click shortcut to COMLNK.
Click Properties in pop-up menu.
Click Program tab in Properties window.
Put a check in box labeled Close on exit.
Click Change icon.
Locate and click satellite dish antenna icon. ⇨
Click OK (return to Program tab in Properties window).
Click Apply.
Click OK.

It is now possible to double-click the COMLNK icon on the desktop to start COMLNK.

Related Topics: See the help topic on how to run in a DOS window.

SECTION 4

COMMUNICATIONS MODULES

4.1 SIGNAL AND NOISE LEVELS IN A/D CONVERTERS.

After passing through the radio frequency (RF) and intermediate frequency (IF) stages of the receiver, the incoming signal plus noise and interference in a microprocessor-based communications receiver is downconverted to form inphase (I) and quadrature (Q) baseband channels. The I and Q channels are fed into integrate-and-dump samplers whose outputs are quantized. The resulting I and Q baseband digital samples provide the inputs to all of the digital signal processing functions in the receiver, including the various tracking loops and the data demodulator. Figure 4-1 illustrates the configuration of the sampled-data digital receiver.

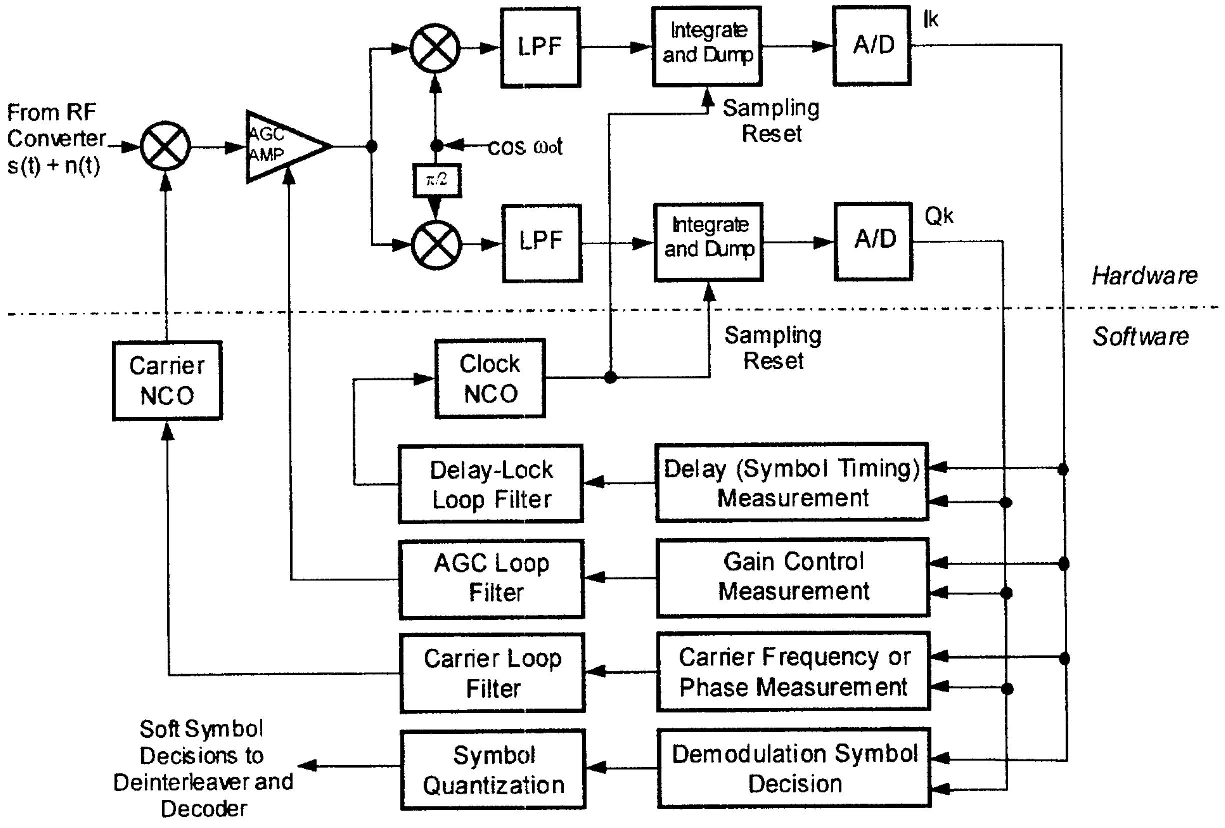


Figure 4-1. Sampled-data digital receiver.

The entire sampling and digitization process depicted in the upper part of this figure is referred to here as analog-to-digital (A/D) conversion. Thus, the A/D converters provide the interface between the signal propagation channel and the receiver digital signal processing functions. It is here where essentially all of the actual simulation of the receiver occurs. The subsequent digital signal processing functions are straightforward software implementations of the same types of algorithms that are commonly implemented in firmware or software.

Simulation of the operation of the A/D converters involves the detailed characteristics of both the channel and the waveform. Consequently, separate A/D converter simulation routines are required for FSK and PSK waveforms. However, certain aspects of the A/D converter simulations are common to all waveforms. These aspects are discussed here.

The following hardware design characteristics are assumed:

a) The nominal value of the rms noise voltage level at the inputs to the A/D converters is a design parameter, specifiable in terms of the resulting quantized value. That is, the nominal rms noise level in the digitized I and Q samples is a specified number of quanta. Note that this is the design level; the channel or a jammer (or imperfect gain control) may vary the actual noise level.

b) An 8-bit gain control word, which may be fixed or varied by a software automatic gain control (AGC) function, multiplies the sampled voltages in the I and Q channels of the A/D converter. The value of the gain control word is in the range from one to 255.

c) The resulting digitized I and Q samples from the A/D converter utilize 16-bit words (15 magnitude bits plus a sign bit in two's complement format). The I and Q samples are in the range from -2^{15} to $+2^{15}-1$.

d) The output of the demodulator, whether a measure of voltage or power, is an 8-bit number in the range from zero to L, where L is a clipping level or a maximum quantization level ($1 \le L \le 255$).

The signal-to-noise ratio (SNR) in the A/D sampling bandwidth at the design point is defined as

$$SNR_o = \frac{C_o}{N_o f_s} \qquad (4.1)$$

where C_o/N_o is the design-point value of the carrier power-to-noise density ratio C/N_o (ratio), N_o is the one-sided noise power spectral density, and f_s is the A/D sampling rate (Hz).

At the design point, the nominal values of the digitized I and Q samples may be represented symbolically as

$$I = G_o[A_o \cos\phi + \text{noise} + \text{interference}]$$
$$Q = G_o[A_o \sin\phi + \text{noise} + \text{interference}]$$

where G_o is the design-point value of the gain control word, A_o is the design-point value of signal voltage amplitude at the A/D sampler, $A_o = \sqrt{C_o}$, and ϕ is an arbitrary phase angle. The standard deviation of the noise voltages in the I and Q channels is

$$\sigma_n = \sqrt{\frac{N_o f_s}{2}} \qquad (4.2)$$

The nominal quantum value of the one-sigma noise level in the digitized I and Q samples is a user-specifiable design parameter, n_o:

$$G_o \sigma_n = n_o \qquad (4.3)$$

The number of noise quanta at the one-sigma level should be large enough that usable low-level signal energy is not lost; the value should be small enough to provide a reasonable dynamic range before strong signals saturate the A/D converter. Here we choose to allow the value of n_o to be specified in the range from one to 255.

From the foregoing equations, the quantum value of the digitized signal voltage at its design-point amplitude is seen to be

$$G_o A_o = n_o \sqrt{2SNR_o}$$

The value of SNR_o is a user-specifiable parameter via the design-point value of C/N_o, represented by C_o/N_o. The value of SNR_o is constrained to be in the range from 0.5 (-3 dB in the sampling bandwidth) to 500 (+27 dB). Thus, the design value of G_oA_o is not less than the noise quantum value n_o, and not more than $32n_o$ quanta.

The actual signal level may be higher or lower than the design-point value. Given a received value of C/N_o, the quantized value of the signal amplitude varies as the square root of the ratio of actual and design values of C/N_o when the gain control word remains fixed at G_o. A signal AGC can be used to control the signal level if desired. The maximum gain change is set by the value of G_o relative to the control range one to 255.

Noise levels may also vary due to background noise and/or interference. A noise AGC can be used to control the noise level if desired. The maximum gain change is again determined by G_o in relationship to the control range.

Default values are provided for the design parameters introduced here, if the user does not specify values. The default values and user-specifiable ranges are summarized in Table 4-1.

Table 4-1. A/D converter parameters.

Parameter	Symbol	Default Value	Allowable Range
1 σ noise quanta	n_o	*	1 to 255
design C/N_o	C_o/N_o	$2f_s$	$f_s/2$ to $500f_s$
design gain	G_o	128	1 to 255
clipping level	L	255	1 to 255

*Note: The default value of the one-sigma noise quanta is set to the nearest integer value of eight times the square root of the number of A/D samples per channel symbol.

4.2 AUTOMATIC GAIN CONTROL.

Some form of gain control is generally required to keep signal levels within the range of A/D quantizers and to maintain tracking loop parameters near their design values. As discussed earlier, the nominal value of the rms noise voltage level at the inputs to the A/D converters is a design parameter, specifiable in terms of the resulting quantized value. The nominal signal level is also a design parameter, specifiable in terms of the design-point signal-to-noise ratio. Because actual signal and noise levels will generally differ from design values, an AGC loop may be used to control them.

The discussion in this section is concerned with operations that are performed using the baseband I and Q digital samples to control the gain of amplifiers or attenuators immediately preceding the A/D converters. AGC operation involving I and Q samples can be noise-based or signal-based. In the absence of any additional noise sources, a noise-based AGC may have little to do. However, when receiver time or frequency tracking errors are large, signal energy may appear to be noise energy, depending on the AGC measurement algorithm. In any event, signal-based AGC operation is generally an important factor in fading channel communications.

Once the AGC measurement is formed in a demodulator, it is fed into a feedback control loop whose implementation does not depend on the measurement algorithm. The AGC loop is implemented in a routine that is used by all of the demodulator modules. Initialization of this routine is handled by a separate routine, which sets up the state variable dataset for a given type of modulation and gain measurement algorithm. Once set up, operation of the AGC loop is controlled by the run-time routine, which provides an 8-bit gain control word that is used as a multiplier in the A/D converters.

The software AGC loop, when switched on, operates in a manner designed to stabilize the signal and/or noise level near the design point with a specified response time. In principle, the AGC measurement can be virtually any function of the gain-controlled signal and noise. Several common types of AGC measurements are outlined later. First, it is appropriate to consider AGC operation from an analytical standpoint to determine how design parameter values are related to loop initialization.

4.2.1 Analysis of AGC Operation.

Digital values of the AGC measurement are accumulated over a specified number of measurements, N. Let x_n denote the n-th value of the AGC measurement x, where the index n ranges from one to N. Then the normalized accumulation, P, is given by

$$P = \frac{1}{\mathrm{N}} \sum_{n=1}^{N} x_n$$

After N measurements, the accumulated value is scaled by N as shown above and then subtracted from a control value, P_o. The resulting difference is accumulated in the AGC accumulator, S:

$$S = \sum (P_o - P)$$

The gain control word, G, is obtained from the most significant eight bits of S:

$$G = \alpha S \qquad , \qquad 1 \le G \le 255$$

where α is a right shift by a bits:

$$\alpha = 2^{-a} \qquad , \qquad \alpha < 1$$

The accumulator S is set initially to $G_o/\alpha + 1/(2\alpha) = 2^a G_o + 2^{(a-1)}$, where G_o is the design-point value of G. Thus, the initial value of G is G_o, and the low-order part of S is set to the middle of its range. In forming the accumulations, neither P nor S is allowed to overflow a 32-bit word. The value of S is further constrained by the allowable range of G.

A relationship between the AGC parameters and the time constant of the AGC loop can be obtained as follows. Consider for the moment a signal AGC where the accumulated measurement P varies with the gain-controlled signal level, GA:

$$P = \left[\frac{GA}{G_o A_o} \right]^{\mu} P_o \qquad , \qquad \text{signal-based AGC}$$

The exponent μ is determined by the measurement algorithm. For an envelope (linear voltage) AGC measurement, $\mu = 1$. For a power (squared voltage) AGC measurement, $\mu = 2$.

To determine the time constant of the AGC feedback control loop via analysis, the AGC accumulator S can be represented by an integral, whereupon the following expression for G is obtained:

$$
\begin{aligned}
G &= \alpha S \\
&= \frac{\alpha}{\Delta t_{AGC}} \int_0^t (P_o - P)\,dt \\
&= \frac{\alpha P_o}{\Delta t_{AGC}} \int_0^t \left[1 - \left(\frac{GA}{G_o A_o} \right)^{\mu} \right] dt
\end{aligned}
$$

where Δt_{AGC} is the AGC loop update interval (*i. e.*, the time between successive values of the AGC accumulation P every Nth measurement). Upon differentiating this expression with respect to time, we have

$$\frac{dG}{dt} = \frac{\alpha P_o}{\Delta t_{AGC}} \left[1 - \left(\frac{GA}{G_o A_o} \right)^{\mu} \right]$$

This equation can be rewritten in the form

$$\frac{dG}{1 - \left(\frac{GA}{G_o A_o} \right)^{\mu}} = \frac{\alpha P_o}{\Delta t_{AGC}} dt$$

Direct integration provides the following result for $\mu = 1$:

$$\frac{G}{G_o} = \frac{A_o}{A} + \left(1 - \frac{A_o}{A} \right) \exp\left(-\frac{\alpha A P_o}{G_o A_o \Delta t_{AGC}} t \right) \qquad , \qquad \mu = 1$$

Similarly, direct integration yields the following result for $\mu = 2$:

$$\frac{G/G_o - A_o/A}{G/G_o + A_0/A} = \left(\frac{1 - A_o/A}{1 + A_0/A} \right) \exp\left(-\frac{2\alpha A P_o}{G_o A_o \Delta t_{AGC}} t \right) \qquad , \qquad \mu = 2$$

The time constant of the AGC response is thus seen to be

$$\tau_{AGC} = \frac{G_o A_o \Delta t_{AGC}}{\mu \alpha A P_o} \tag{4.4}$$

The AGC loop exhibits a highly nonlinear response, and the effective time constant depends on the relative change in signal strength. For small changes in signal level, $A_o / A \approx 1$, the following relationship is obtained for the shift parameter α as a function of the AGC design values:

$$\alpha = \frac{G_o \Delta t_{AGC}}{\mu P_o \tau_{AGC}}$$

The AGC accumulator S resides in a 32-bit word, and the gain control word G occupies the most significant eight bits of S. To avoid any potential problem with the sign bit in a Fortran language implementation, no more than 31 bits should be used for the accumulator. Therefore, it is necessary that the shift count satisfy the following requirement:

$$\begin{aligned} a &= -\log_2 \alpha \\ &\leq 23 \end{aligned} \tag{4.5}$$

Another constraint is imposed by the requirement for loop stability. To ensure that the sampled-data AGC loop is stable, it should be updated at a rate not less than 10 times the effective closed-loop bandwidth. The bandwidth of a linearized first-order loop is equal to the reciprocal of four times the loop time constant. Consequently, the stability requirement can be stated as

$$\Delta t_{AGC} \leq 4\tau_{AGC}/10 \tag{4.6}$$

The significance of these two requirements on the implementation of the digital AGC loop is discussed in the next subsection.

4.2.2 AGC Measurement.

The gain control measurement is necessarily specialized to the modulation waveform and hence is different for FSK and PSK receivers. However, to determine the significance of the foregoing requirements, it is necessary to examine a specific type of AGC measurement. Consider, for example, a power measurement in the demodulator matched filter:

$$x = \left[\frac{1}{K}\sum_{k=1}^{K} I_k\right]^2 + \left[\frac{1}{K}\sum_{k=1}^{K} Q_k\right]^2$$

where k is a sampling index within the demodulated symbol period, I_k and Q_k are digital samples from the A/D converter, and K is the number of A/D samples per symbol period.

The AGC control value P_o is equal to the mean value of the AGC measurement when the gain-controlled signal plus noise is at the design point. For the above power measurement, the mean value at the design point can be shown to be

$$<x> = (G_o A_o)^2 + 2(G_o \sigma_n)^2/K$$

where σ_n is the standard deviation of the noise voltage in the I and Q channel digital samples (Eq. 4.2). Using the design-point values of the signal and noise components of these samples (see the discussion of the A/D converter in Section 4.1), we have

$$<x> = 2n_o^2(SNR_o + 1/K)$$

where n_o is the nominal value of one-sigma noise quanta in the digital I and Q samples (Eq. 4.3), and SNR_o is the design-point signal-to-noise ratio (Eq. 4.1).

The requirement on the shift count a (Eq. 4.5) imposes a lower limit of 2^{-23} on the magnitude of α. To investigate the effect of this limit when the AGC control value, P_o, is given by the above expression for <x>, take G_o at its smallest value of unity, n_o at its largest value of 255, and SNR_o at its largest value of 500. Choose $K = 1$ and note that $\mu = 2$ in this example. The ratio $\Delta t_{AGC}/\tau_{AGC}$ can be increased if necessary by reducing the AGC update rate as long as the stability criterion (Eq. 4.6) is satisfied, so this ratio can be taken at its largest value of 0.4. With these extreme values, the foregoing equations yield a value of α of 2^{-28}. This is unacceptably small, therefore some type of scale factor must be introduced.

Examination of the above equation for <x> suggests that the AGC measurement be divided by the design value of noise quanta in the A/D samples, n_o. To enable division to be replaced by a binary shift operation, we scale instead by $2^{\log_2 n_o}$. The logarithm is rounded up if n_o is not a power of two. This scaling increases the minimum value of α in the present example to 2^{-20}, which is within the allowable range. Scaling power measurements in this manner also has the desirable property of limiting the number of noise quanta in the low-order bits of the measurements near the design value.

The other requirement on α is that it must be less than unity. To investigate this limit with P_o given by <x> above, take G_o at its largest value of 255 with n_o at its smallest value of unity. Take SNR_o at its smallest value of 0.5 and set the ratio $\Delta t_{AGC}/\tau_{AGC}$ at its largest value of 0.4 with $\mu = 1$ and K large. We now find that α must have a value of 102, which is unacceptable. The solution here is to update the AGC loop more frequently, if possible, to reduce the value of Δt_{AGC}. If Δt_{AGC} cannot be reduced, then the AGC time constant τ_{AGC} must be increased. Alternatively, the user may choose to increase either or both of the values of n_o and SNR_o until $\alpha < 1$.

4.2.3 AGC Initialization.

Initialization of the AGC loop is dependent on the modulation waveform, the type of measurement, and several design parameters discussed above. AGC design parameters include the loop update rate, time constant, design-point voltage gain factor, and design-point noise level.

With the convention that AGC power measurements are scaled by dividing them by $2^{\log_2 n_o}$ while envelope and coherent measurements are not scaled, the control value P_o is listed in Table 4-2 for several common types of AGC loops. The parameter K is the number of samples that are coherently summed per symbol, and M is the number of M-ary FSK signaling states. All sums over K are assumed to be scaled by dividing by K.

Table 4-2. AGC control values.

Type of AGC Loop	AGC Control Value
FSK, maximum filter	$P_o = 2n_o(SNR_o + 1/K)$
FSK, sum filters	$P_o = 2n_o(SNR_o + M/K)$
FSK, noise filters	$P_o = 2n_o(M-1)/K$
FSK, baseband power	$P_o = 2n_o(SNR_o + 1)$
PSK, total power	$P_o = 2n_o(SNR_o + 1/K)$
PSK, coherent signal envelope	$P_o = n_o\sqrt{2\,SNR_o}$
PSK, noise power	$P_o = 2n_o/K$

The last expression in this table assumes that the noise power measurement in a PSK demodulator is made by squaring the difference of samples obtained in the first and second halves of the symbol period in both the I and Q channels.

In accordance with the scaling discussed above, the value of n_o in these expressions is to be interpreted as $n_o^2/2^{\log_2 n_o}$ when a power measurement is involved. This allows for the fact that n_o is not necessarily a power of two, while scaling is accomplished via a binary shift operation.

Expressions for P_o can also be obtained for FSK or PSK envelope detection, but they are considerably more complicated. As indicated by the next to last expression listed in the table, a rough approximation is formed by taking the square root of the factor that multiplies n_o in the expressions for power detection. The consequence of using an inexact value of the control value P_o is that the AGC gain stabilizes at a value somewhat different than the nominal design-point value G_o. The AGC initialization routine, IAGC, has been tuned to provide AGC stabilization quite close to the design value in all cases. The AGC loop itself is implemented in subroutine AGC.

4.2.4 Summary of AGC Design Parameters.

Each receiver provides the option of an AGC loop that the user may enable or disable. The loop can be made to operate as either a signal-based AGC or a noise-based AGC. A number of design parameters can be specified, including the design-point noise level, the design-point voltage gain factor, the AGC time constant, and the number of gain measurements per loop update.

The AGC design-point noise level is the one-sigma value of the noise quanta in the A/D outputs when the signal-to-noise ratio is equal to the design-point value of C/N_o, and the AGC gain is at its design value. Values from one to 255 may be specified. Large values resolve weak signals but can cause quantizer saturation with strong signals. Small values minimize saturation but can cause bottoming in weak signal conditions.

The AGC design-point voltage gain factor establishes the AGC dynamic range in both directions. The AGC provides an 8-bit gain control word to the A/D converters. When the signal is at its design level, the gain control word has the value specified. Values from one to 255 are acceptable. A value of 255 causes the AGC to operate only as a variable attenuator, with no ability to respond to a decrease in input level. The other extreme value, unity, provides no ability to respond to an increase in signal level. An intermediate value allows the voltage gain to increase or decrease in the range from one to 255. If a value of 64 is entered, for example, the gain can increase by +12 dB and decrease by -36 dB. If the AGC is turned off, the specified gain factor cannot change.

The AGC time constant determines the speed at which the AGC responds to changes in incoming signal or noise level. For fading channel operation, it is recommended that a signal-based AGC with a fairly large time constant, or a noise-based AGC, be employed. The objective is to avoid boosting the noise level when the signal fades.

The number of AGC measurements per loop update determines the loop iteration rate. Between iterations, the measurements are accumulated, which amounts to pre-smoothing. The specified value is used unless it adversely affects loop stability. In those cases, it will be changed by the program at run time to ensure that the digital AGC loop is stable.

The AGC measurement algorithms for both noise-based and signal-based operation are necessarily tailored to the modulation waveform. Several examples of AGC measurement algorithms can be found in the demodulator code listings (see the source code for modules MFSK1, MFSK2, MPSK1, MPSK2, and MPSK3). Descriptions of these and other AGC algorithms employed in the demodulator modules are available in the literature; see, for example, (Bogusch, 1990).

4.3 DELAY-LOCK LOOP.

Variation in received signal time delay can occur because of oscillator drift, transmitter and receiver motion, and total electron content (TEC) variation. Active tracking of signal delay is usually required to maintain synchronization of spreading codes and to align demodulator A/D samples with symbol timing. This is normally accomplished with a delay-lock loop (DLL).

A DLL is functionally identical to a phase-lock loop (PLL). The only difference is in the algorithms used to form the respective error measurements. Thus, the DLL in COMLNK is implemented using the same routine used for the PLL. The user can specify first-order, second-order, or third-order DLL operation. As for the PLL, either a multiply-and-shift implementation or a shift-only implementation can be specified for the loop filter.

The configuration of the sampled-data loop filter is illustrated in Figure 4-2. The delay error measurement is input to the filter at a specified update rate. The output of the filter advances or retards the phase of the clock numerically-controlled oscillator (NCO) to maintain proper timing. A third-order loop has nonzero values for all three filter coefficients. A second-order loop is obtained by setting C_3 to zero, while a first-order loop results from setting both C_3 and C_2 to zero. The values of the loop filter coefficients depend on the user-specifiable parameters discussed below. Expressions for the coefficients are presented in (Bogusch, 1990).

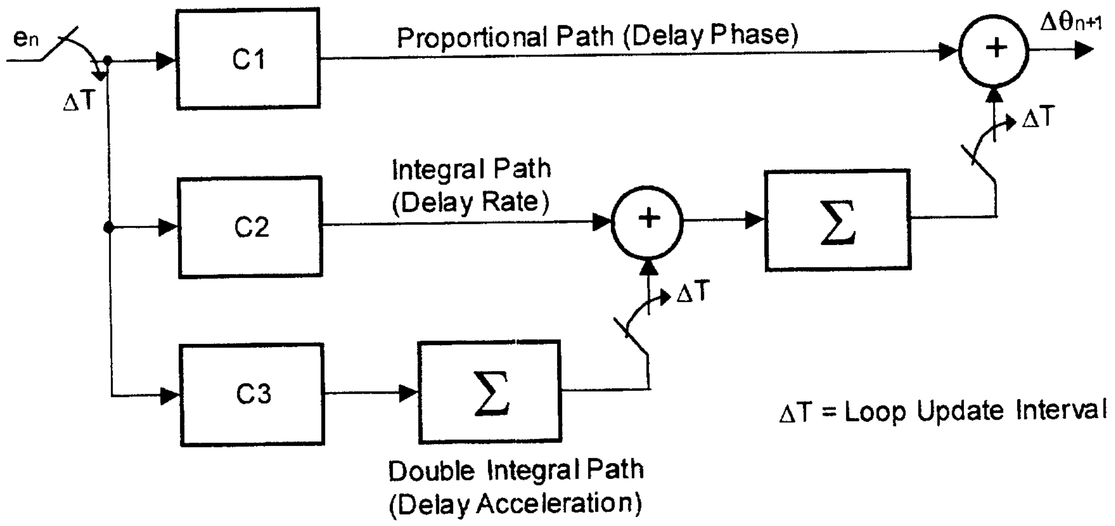


Figure 4-2. Sampled-data loop filter.

User-specifiable design parameters include the loop noise bandwidth, the damping factor (for second-order loops), and the direct path gain and gain margin (for third-order loops). The loop noise bandwidth determines the speed at which the DLL responds to changes in received signal timing caused by range or TEC dynamics, clock oscillator drift, or frequency selective effects. Note that the specification is the noise bandwidth, not the natural frequency. The two parameters are related in a manner that depends on the loop order. For fading channel operation, it is recommended that as small a DLL bandwidth be employed as dynamics requirements permit. The objective is to avoid responding to noise in fades, which can cause loss of lock.

The damping factor controls the transient response of a second-order loop. Small values provide rapid response with large overshoot; large values provide slower response with less overshoot. Indeed, a second-order loop approaches a first-order loop when the damping factor is very large. Critical damping is obtained with a damping factor of unity. The classic Jaffe-Rechtin loop filter is obtained with a damping factor of 0.707. Either value is a good choice, with the latter supplied as the default value.

For third-order loops, the damping factor is replaced by two other parameters that determine speed of response and stability. The direct path gain must not be less than unity, and the value is two when the denominator polynomial of the closed loop transfer function has a Butterworth form. An optimum value for this parameter that maximizes loop stability and minimizes the ratio of loop noise bandwidth to the natural frequency is supplied automatically if you enter zero.

The gain margin determines how far the input signal level can drop below the design level before the third-order loop becomes unstable. In decibels, the margin is $20\log_{10}$ of the value given as input. A gain margin of four (12 dB) corresponds to a Butterworth form for the denominator polynomial of the closed loop response. For increased stability in fading channels, the program default is a gain margin of ten (20 dB).

The number of DLL measurements per loop update determines the loop iteration rate. Between iterations, the measurements are accumulated, which amounts to pre-smoothing. The value specified as input will be used unless it adversely affects loop stability. In that case, it will be changed by the program at run time to ensure that the sampled-data loop is stable.

The DLL, like the other tracking loops in COMLNK, is a pure integer arithmetic design. Subroutine PLLP implements the loop filter using binary shift and integer multiplication operations. Subroutine PLLQ implements the loop filter using shift operations exclusively. These two routines are in the code module named PLL. Note that no floating-point operations are involved. The code is intended to be suitable for use in microprocessor-based equipment. The necessary scaling for the DLL is taken care of automatically by an initialization routine, IPLL, which initializes the state variable dataset. This routine uses subroutine ICOEF to convert the loop filter coefficients to integer format. The resulting scale factors and loop parameters are provided in the print output.

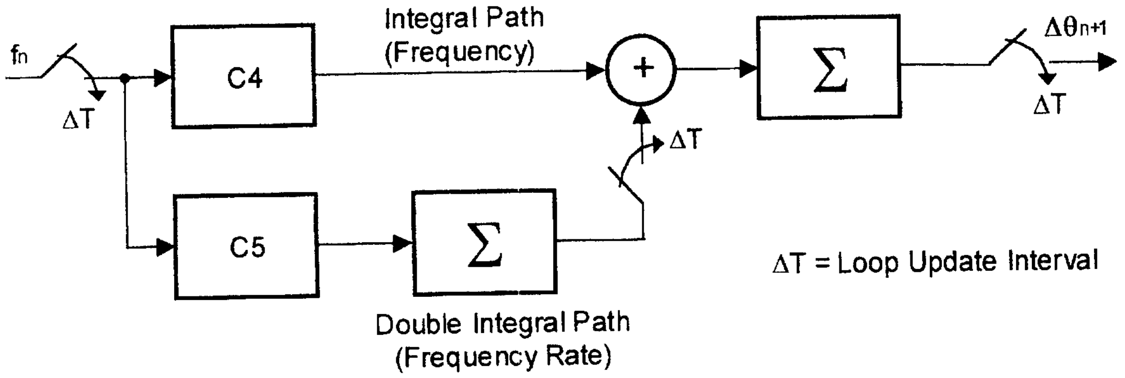
4.4 FREQUENCY-LOCK LOOP.

Variation in the received carrier frequency may result from oscillator drift and Doppler shift due to transmitter and receiver motion. Systems that utilize transionospheric propagation paths may also experience Doppler-like frequency shifts due to variation in the ionospheric TEC. In any case, active tracking of the received carrier frequency is normally required to keep the signal centered in the bandpass of receiver filters. This is usually accomplished using a frequency-lock loop (FLL), also referred to as automatic frequency control (AFC).

Even when phase-lock loop (PLL) carrier tracking is used for coherent demodulation, use of FLL techniques may be required for frequency acquisition. A PLL by itself is a poor frequency acquisition circuit when the initial frequency uncertainty is larger than the PLL bandwidth. On the other hand, an FLL can readily acquire initial frequency offsets that greatly exceed the FLL bandwidth.

FLL measurement error algorithms (frequency discriminators) depend on the type of waveform. Several discriminator algorithms are described in (Bogusch, 1990). The FLL itself is independent of the measurement technique and hence is implemented in a separate subroutine that is used by all of the demodulators.

Figure 4-3 shows the configuration of the loop filter for the sampled-data FLL. Except for the fact that there is no proportional path in the FLL filter, the implementation is similar to that of a DLL or PLL. The user can specify a first-order or second-order FLL. As for the DLL and PLL, either a multiply-and-shift implementation or a shift-only implementation can be specified.



f_n
ΔT
C4
Integral Path (Frequency)
+
Σ
$\Delta\theta_{n+1}$
ΔT
ΔT
C5
Σ
ΔT = Loop Update Interval
Double Integral Path (Frequency Rate)

Figure 4-3. Loop filter for sampled-data FLL.

User-specifiable design parameters include the loop noise bandwidth and the damping factor (for second-order loops). The loop noise bandwidth determines the speed at which the FLL responds to changes in received carrier frequency caused by range or TEC dynamics, clock oscillator drift, or signal scintillation. Note that the specification is the noise bandwidth, not the natural frequency. The two parameters are related in a manner that depends on the loop order.

For fading channel operation, it is recommended that as small a bandwidth be employed in the FLL as dynamics requirements permit. This reduces the danger of losing lock due to noise in fades or reacting to the Doppler noise produced by signal scintillation.

The damping factor controls the transient response of a second-order loop. Small values provide rapid response with large overshoot; large values provide slower response with less overshoot. Indeed, a second-order loop approaches a first-order loop when the damping factor is very large. Critical damping is obtained with a damping factor of unity. The classic Jaffe-Rechtin loop filter is obtained with a damping factor of 0.707. Either value is a good choice, with the latter supplied as the default value.

The number of FLL measurements per loop update determines the loop iteration rate. Between iterations, the measurements are accumulated, which amounts to pre-smoothing. The value specified as input will be used unless it adversely affects loop stability. In that case, it will be changed by the program at run time to ensure that the sampled-data loop is stable.

A frequency lock detector is supplied with the FLL in COMLNK. The algorithm used to implement the lock detector is that described in (Bogusch, 1990). The dot product of I and Q channel samples is smoothed in a digital single-pole filter to form the frequency lock estimate. The filter time constant controls the response speed of the lock detector. To minimize false loss-of-lock detections that could trigger reacquisition logic, it is suggested that the FLL lock detector filter time constant be larger than the largest value of scintillation decorrelation time.

A threshold level is used to test the detector output to detect acquisition of frequency lock. A lock loss threshold is set at one-half the lock level to provide a null zone or hysteresis effect to prevent levels near the threshold from producing multiple indications of lock and loss-of-lock. To facilitate specification of the lock threshold it is entered on a scale of zero to one, where unity corresponds to lock on a design-level signal without noise. It is suggested that a value near 0.5 be specified. The input value is converted internally to an appropriate integer using the specified design signal level.

The FLL, like the other tracking loops in COMLNK, is a pure integer design. Subroutine FLLP implements the loop filter using binary shift and integer multiplication operations. Subroutine FLLQ implements the loop filter using shift operations exclusively. These two routines are in the code module named FLL. No floating-point operations are involved. The code is intended to be suitable for use in microprocessor-based equipment. The necessary scaling for the FLL is taken care of automatically by an initialization routine, IFLL, which initializes the state variable dataset. This routine uses subroutine ICOEF to convert the loop filter coefficients to integer format. The resulting scale factors and loop parameters are provided in the print output.

4.5 PHASE-LOCK LOOP.

Coherent PSK demodulation requires reconstruction of a local carrier phase reference. This is usually accomplished using some type of suppressed-carrier tracking loop, typically a power-law, Costas, or modified Costas loop. While the algorithm for phase error measurement depends on the type of loop and the waveform (*e.g.*, BPSK, QPSK, OQPSK), implementation of the phase-lock loop (PLL) itself is independent of such details.

Thus, the PLL is implemented in a subroutine that is invoked for each waveform. Furthermore, a PLL is functionally identical to a delay-lock loop (DLL). The method of forming error measurements and initialization of the state variable dataset varies with the waveform and type of loop, but the same run-time loop filter routine is used for both PLL and DLL operation.

The user can specify first-order, second-order, or third-order PLL operation. Either a multiply-and-shift implementation or a shift-only implementation can be specified. The configuration of the loop filter is identical to that shown in Figure 4-2.

Other user-specifiable design parameters include the loop noise bandwidth, the damping factor (for second-order loops), and the direct path gain and gain margin (for third-order loops). The loop noise bandwidth determines the speed at which the PLL responds to changes in received carrier phase caused by range or TEC dynamics or by signal scintillation. Note that the specification is the loop noise bandwidth, not the natural frequency. The two parameters are related in a manner that depends on the loop order (Bogusch, 1990).

In a fading channel, loss of phase lock or cycle slipping is generally inevitable. If the channel symbol rate times the smallest value of channel decorrelation time τ_0 is below about 40 to 100, it is recommended that coherent demodulation not be used. Otherwise, a large PLL bandwidth is recommended to track signal phase scintillation.

The damping factor controls the transient response of a second-order loop. Small values provide rapid response with large overshoot; large values provide slower response with less overshoot. A second-order loop approaches a first-order loop when the damping factor is very large. Critical damping is obtained with a damping factor of unity. The classic Jaffe-Rechtin loop filter is obtained with a damping factor of 0.707. Either value is a good choice, with the latter supplied as the default value.

For third-order loops, the damping factor is replaced by two other parameters that determine speed of response and stability. The direct path gain must not be less than unity, and the value is two when the denominator polynomial of the closed loop transfer function has a Butterworth form. An optimum value for this parameter that maximizes loop stability and minimizes the ratio of loop noise bandwidth to the natural frequency is supplied automatically if you enter zero.

The gain margin determines how far the input signal level can drop below the design level before the third-order loop becomes unstable. In decibels, the margin is $20\log_{10}$ of the value given as input. A gain margin of four (12 dB) corresponds to a Butterworth form for the denominator polynomial of the closed loop response. For increased stability in fading channels, the program default is a gain margin of ten (20 dB).

The number of PLL measurements per loop update determines the loop iteration rate. Between iterations, the measurements are accumulated, which amounts to pre-smoothing. The value specified as input will be used unless it adversely affects loop stability. In that case, it will be changed by the program at run time to ensure that the sampled-data loop is stable.

A phase lock detector is supplied with the PLL in COMLNK. The algorithm used to implement the lock detector is that described in (Bogusch, 1990). The difference between inphase and quadrature channel samples is smoothed in a digital single-pole filter to form the phase lock estimate. The filter time constant controls the response speed of the lock detector. When adaptive CPSK/DPSK demodulation or adaptive PLL/FLL switching is enabled, rapid response depends on the use of a relatively small lock detector time constant.

A threshold level is used to test the detector output to detect acquisition of phase lock. A lock loss threshold is set at one-half the lock level to provide a null zone or hysteresis effect to prevent levels near the threshold from producing multiple indications of lock and loss-of-lock. To facilitate specifying the lock threshold it is entered on a scale of zero to one, where unity corresponds to lock on a design-level signal without noise. It is suggested that a value around 0.3 to 0.5 be specified. The input value is converted internally to an appropriate integer using the specified design signal level.

The PLL, like the other tracking loops in COMLNK, is a pure integer arithmetic design. Subroutine PLLP implements the loop filter using binary shift and integer multiplication operations. Subroutine PLLQ implements the loop filter using shift operations exclusively. These two routines are in the code module named PLL. No floating-point operations are involved. The code is intended to be suitable for use in microprocessor-based equipment.

The necessary scaling for the PLL is taken care of automatically by an initialization routine, IPLL, which initializes the state variable dataset. This routine uses subroutine ICOEF to convert the loop filter coefficients to integer format. The resulting scale factors and loop parameters are provided in the print output.

4.6 USER BIT SOURCE AND SINK.

All simulations involve the transmission of a stream of digital data, obtained by processing the user bit stream from a specified source. Two basic options are available for specifying the source data: an arbitrary but fixed bit stream, or a pseudorandom bit stream.

A fixed source is contained in a file whose name is specified by the user. The contents of the file can be a text message that employs the American Standard Code for Information Interchange (ASCII) character set, or it may be a sequence of hexadecimal values. If an ASCII message is specified, the user can choose between a 6-bit subset code, the standard 7-bit code, or the 8-bit extended ASCII code. If a hexadecimal source is chosen, the full 32-bit source words are used.

If a fixed source file is specified, the program looks for the file in the current directory. If it is not found there, the program looks for the file in the directory that contains the layout data file. If the file still is not found, the program automatically substitutes a pseudorandom source.

The transmitted message is formed by extracting the indicated number of data bits (6, 7, 8, 32) from each character or word in the file, up to a specified message length. Any message length, subject to available memory, may be used. If the message length is specified as zero, the entire file is used. If the file is ASCII text, a zero-length specification also causes a carriage return, line feed character pair to be appended at the end of each line of text. This enables formatted messages to be handled properly at the end of the link. The resulting bit stream is repeated as needed to achieve the specified number of user bits, which are transmitted through the link.

If a pseudorandom source is selected, a random number generator is used to form 32-bit pseudorandom words. The starting generator seed is an input. All 32 bits from each pseudorandom word are extracted and processed for transmission.

With any type of source, the maximum number of user bits and the maximum number of bit errors are specified as part of the source data. These values determine the simulation run length. The run will automatically terminate when the total number of received user bits reaches the specified maximum value, or when the user bit error count reaches the specified maximum value, whichever occurs first.

Any number of user bits can be specified, up to 2^{31} - 1. All counters use 32-bit signed words, for which the largest positive value is $2^{31} - 1 = 2,147,483,647$. In most cases, use of techniques such as coding, repetition, and multiplexing cause the channel bit rate to exceed the user bit rate. Therefore, the channel bit counter (and perhaps other intermediate counters as well) will likely saturate when the maximum number of user bits is specified. No counters will overflow, and all counts will be correct up to and including the point of saturation.

Any number of errors, one or more, can be specified. A specification of zero errors causes the program to substitute the total number of bits. This forces the run to continue until the specified number of bits has been received.

If a fading channel is involved, the run length should be chosen to encompass several thousand decorrelation times or more to achieve reasonable statistical significance. If two or more fading channels are in the layout, the largest value of decorrelation time should be used in setting the run length.

Other inputs to the source module include the user block size and packet sizes. The number of bits per user block is used only if the layout does not use a CRC or Hamming code. Otherwise, the block size is set by the CRC or Hamming block. Up to five packet sizes can be specified; the number of bits per packet is used to gather packet error statistics. Zero-length packets are ignored.

A precise definition of the manner in which the user bit stream is formed can be obtained by examining the listing of the source code module, subroutine SOURCE. As in all COMLNK run-time modules, an initialization routine sets up the run-time routine. The source initialization routine, ISOURC, sets up the source state variable dataset during an initial linking procedure that immediately precedes simulation execution. The state variable dataset for each module is passed to the module by the interrupt controller via the module call sequence.

The corresponding sink module at the end of the link, implemented in subroutine SINK with initialization routine ISINK, performs a bit-by-bit comparison of the received user bits with the transmitted bit stream. Counts of user bit errors, user or CRC or Hamming block errors, and packet errors for up to five packet sizes are accumulated for subsequent output, along with counts of the total numbers received and the average error rates.

The sink module also provides optional on-screen display of received messages. For ASCII messages, the screen display can be enabled for all received text, or only those lines that contain errors can be displayed. Errors are displayed in red on the screen. If a hexadecimal or random source is specified, errors in the received user bits are displayed in the form of a hexadecimal error syndrome (1 denotes an error in a given bit position, 0 denotes no error). In either case, the screen display can be disabled if desired. The user may also choose to have the sink module write the received text or error syndrome to a file.

4.7 CYCLIC REDUNDANCY CHECK.

A cyclic redundancy check (CRC) block code is used to detect any uncorrected errors in the received user data. A CRC encoder appends a parity tail at the end of each transmitted data block so that a CRC decoder can check the parity bits at the receiver to detect the presence of errors.

The number of parity bits, P, and the number of data bits per block, D, are user-specifiable design parameters. The parity tail may consist of one to 32 bits, and there may be any number of data bits in each block. Note that the specified user bit rate is the rate at which the parity-encoded block is transmitted. Because of the insertion of parity bits, the actual rate at which user data bits are transmitted is reduced by the ratio D/(D+P).

Parity bits are generated by feeding the user data bits into a linear feedback shift register (LFSR). Feedback tap connections are established by a CRC parity generator polynomial, which is a user-specifiable design parameter. The generator polynomial is simply a bit pattern; a bit set to "1" denotes a connection to the corresponding stage of the LFSR, while "0" denotes no connection. The generator may contain any number of nonzero bits up to the specified number of parity bits. Being a bit pattern, it is conveniently displayed in hexadecimal format. The generator polynomial is referenced to the shift register output.

The user can also specify the initial contents of the LFSR, *i. e.*, the bit pattern that is loaded into the register at the start of each block. Any value containing up to the number of parity bits can be specified. A zero value can be given but is not recommended because it allows erroneous all zero blocks to go undetected. In a fading channel, an all-zero error pattern can result from quantizer bottoming. Such errors can be detected by using a nonzero initial register preset, such as an all ones pattern.

A CRC encoder can be implemented in several ways. The implementation used here is shown in Figure 4-4 (Bogusch, 1989b). This example is for a 16-bit CRC with generator polynomial of 1021 (hex). Each data bit is modulo-two added to the contents of the shift register stages defined by the generator polynomial, and the result is shifted into the register. After all data bits in the block have been input in this manner, the feedback connection is opened and zeros are shifted into the register. The parity bits then appear at the output of the multiple-input modulo-two adder and are appended to the data bits.

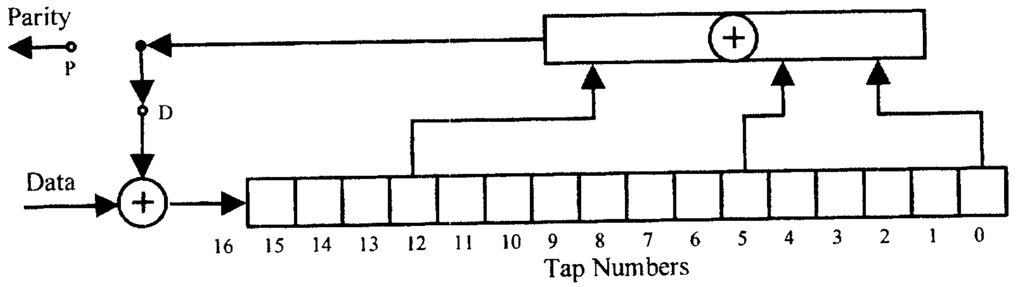


Figure 4-4. CRC block encoder.

Whereas the feedback connection in the encoder is opened after the data bits are fed into the LFSR, in the decoder the feedback connection remains closed. This causes the received parity bits to be modulo-two combined with those generated in the decoder. If no errors are present, the two parity tails match and the resulting modulo-two sum (referred to as the error syndrome) is zero. Errors in the received data or parity bits produce one or more mismatches that result in a nonzero syndrome. The syndrome is passed to the sink module at the end of each received CRC block and used to count CRC block errors.

The CRC encoder is implemented in run-time routine CRCCOD, with initialization routine ICRCOD. The decoder is implemented in run-time routine CRCCHK and has the same initialization routine.

4.8 HAMMING CODE.

A Hamming code is a relatively high-rate binary code in which a set of parity check bits is appended to a block of data bits, which are not changed by the encoder. Examples include (15,11) and (31,26) codes. The first number is the size of the code block in bits, and the second is the number of data bits per block. The difference is the number of parity bits. The limited redundancy in the code block means that Hamming codes are capable of correcting only a single error in each block. Typically, an extra parity bit is appended so that the code is capable of detecting (but not correcting) the presence of two errors in a block. The resulting single-error correcting, double-error detecting codes are called extended Hamming codes. Examples include (16,11) and (32,26) codes.

The extremely weak error-correction capability of Hamming codes makes them poorly suited for use in fading channels or in any link wherein errors may occur in bursts. While the error-correction capability is best left unused, the error-detection capability can be useful. Extended Hamming codes can detect most combinations of an even number of errors and all combinations of an odd number of errors. Although CRC codes provide more powerful error-detection capability, Hamming codes are still found in some applications. A notable example is a variation of the (32,26) code used to encode Global Positioning System (GPS) navigation data.

In the implementation provided here, parity bits are appended at the end of each block, following the data bits. The internal register format for the (32,26) code used with the GPS NAV data is illustrated in Figure 4-5.

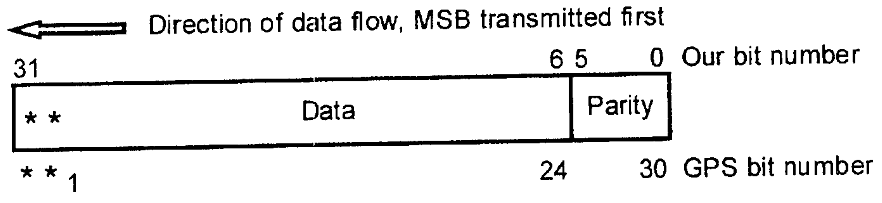


Figure 4-5. Hamming encoder register format for GPS.

The left-most bits, shown as **, are bits 29 and 30 from the previous encoded NAV data word. The six right-most bits are the parity bits generated for the current word. Words 1, 2, and 10 in each subframe are encoded such that bits 29 and 30 are zeros, with bits 23 and 24 used to provide proper parity.

It is seen that the (32,26) code used in GPS links together 30-bit words. The parity-encoding algorithm causes bits 1-24 of each word to be inverted if bit 30 of the previous word is a one. In effect, the last parity bit in each word is used to differentially encode the data bits in the following word. The enables a polarity ambiguity resulting from phase inversion in coherent PSK demodulation to be resolved by the Hamming decoder.

Another peculiarity of the GPS variation of the (32,26) Hamming code is that there are effectively two block sizes insofar as the encoder is concerned. In each subframe consisting of ten 30-bit words, the first two and last are encoded with 22 actual data bits, with the six parity bits placed in positions 23 through 28 and two zero bits placed in positions 29 and 30. The other seven words in each subframe are encoded with 24 data bits, with the six parity bits placed in positions 25 through 30. This enables some of the encoding to be performed on the ground and some to be performed onboard the satellite. While this complicates the encoder, the result is completely transparent to the decoder. The GPS Hamming decoder treats all words as if they had 24 data bits.

The implementation of the Hamming encoder and decoder provided here faithfully reproduces the algorithms used in GPS but is not restricted to the GPS format. Indeed, this encoder and decoder can be used for conventional (15,11) and (31,26) Hamming codes as well as for normal extended (16,11) and (32,26) codes. All five options are built in and are switch-selectable from the user interface.

In addition, the user can customize the Hamming encoder using an extension of the features needed to accommodate the GPS format. The number of parity bits can be specified in the range from four to six. Two block sizes, each with a separate set of parity generator coefficients, can be specified. The number of blocks in an encoding cycle can be specified in a range from zero to 32, and the pattern of short and long blocks in the cycle can be specified. The data inversion option can be turned on or off. The generator coefficients must be specified by the user when a customized Hamming code is thus created.

In all cases, use of the Hamming code reduces the rate at which source bits are transmitted by the ratio D1/(D2+K), where D1 is the number of data bits in a short block, D2 is the number of data bits in a long block, and K is the number of parity bits per block.

Parity bits are generated by modulo-two addition of a set of data bits. The bit positions included in the formation of each parity bit are specified by the code generator coefficients. Each coefficient is simply a bit pattern, wherein a one indicates that the bit in that position is to be included in the modulo-two sum. Being bit patterns, the coefficients are conveniently displayed in hexadecimal format.

The Hamming decoder modulo-two combines the received parity bits with those it generates internally from the received data bits. If there are no errors the two parity tails match and the resulting modulo-two sum (referred to as the error syndrome) is zero. Errors in the received data or parity bits normally produce one or more mismatches that result in a nonzero syndrome. The syndrome is passed to the sink module at the end of each received Hamming block and used to count Hamming block errors.

The Hamming encoder is implemented in run-time routine HAMMNX, with associated initialization routine IHAMMN. The decoder is implemented in run-time routine HAMMDR, with initialization routine IHAMMD.

4.9 DIFFERENTIAL ENCODING.

When coherent phase-shift keyed (CPSK) modulation is used in conventional unhopped links, phase ambiguities inherent in suppressed-carrier phase tracking must be resolved. For example, in a biphase-shift keyed (BPSK) system, the demodulator phase tracking loop can lock with equal probability in the correct phase state around zero error, or in an ambiguous phase state around 180 degrees phase error. In the absence of other errors, the latter condition causes all demodulated bits to be inverted. That is, all zeros become ones and all ones become zeros. A method of rendering the receiver insensitive to such phase inversions is therefore required.

Differential encoding of the transmitted data, combined with differential decoding of the received data, satisfy this requirement. Differential encoding operates on a binary data sequence to produce a different sequence wherein the data are contained in binary transitions, not in the binary values themselves. For example, an input data bit of zero causes the differentially encoded output sequence to retain the same state, regardless of whether that state is zero or one. An input data bit of one causes the differentially encoded sequence to change states, regardless of the preceding state.

In the receiver, the differential decoder performs the reverse operation. Thus, no change in input states is transformed to an output 0, while an input state change becomes an output 1. In this manner, the correct data are produced (in the absence of demodulation errors) no matter in which phase state the biphase suppressed-carrier loop may be locked. Both the encoding and decoding operations are shown in Figure 4-6 where T denotes a delay of one bit period, and the other operation is an exclusive-or (modulo-two addition).

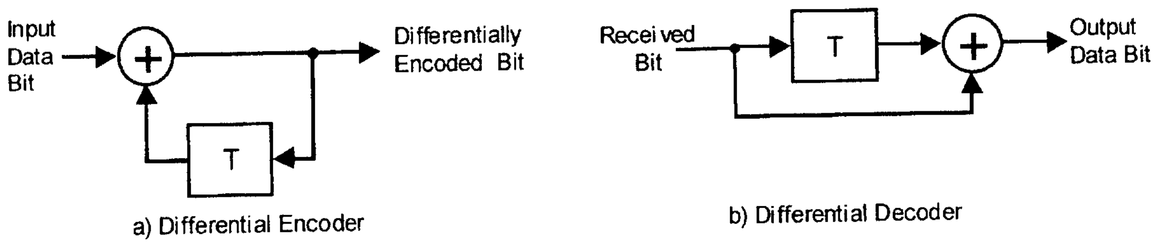


Figure 4-6. Differential encoder/decoder.

The differential encoder is implemented in run-time routine DIFENC, with initialization routine IDFENC. The decoder is implemented in run-time routine DIFDEC and has the same initialization routine.

4.10 BINARY TO M-ARY CONVERSION.

When more than two signaling states are employed with frequency-shift keyed (FSK) modulation, binary data must at some point be converted to the signaling alphabet. For example, with 4-ary (quaternary) FSK, two bits are transmitted per modulation symbol. With 8-ary FSK, three bits are transmitted per symbol. While any number of signaling states can, in principle, be employed, most systems transmit an integral number of bits per symbol. Therefore, COMLNK implements M-ary modulation where M is a power of two from two to 128 (one to seven bits per symbol).

In the transmitter chain, a binary-to-M-ary converter performs the function of converting binary symbols to M-ary symbols. The implementation simply involves shifting input bits into a symbol register. When the specified number of bits per symbol has been inserted, the resulting M-ary symbol is output. The M-ary output symbol rate is reduced from the input bit rate by a factor of $\log_2 M$.

The complementary operation is performed by an M-ary-to-binary converter in the receiver chain. Each input M-ary symbol is broken down into its component bits, and thus the output bit rate is increased from the input symbol rate by a factor of $\log_2 M$.

When binary error-correction encoding with soft-decision decoding is employed in an M-ary FSK link, the operation of the M-ary-to-binary converter is complicated by the requirement to convert soft M-ary symbol metrics from the demodulator to soft binary symbol metrics required by the decoder. A number of algorithms can be used for this purpose. The one most commonly encountered in practice and used in COMLNK is that described by (Bogusch, 1989b).

The soft metric conversion algorithm involves determining the maximum competing M-ary metrics for the two binary possibilities in each bit position of the M-ary symbol. When dual-metric decoding is selected (see the discussion of the binary convolutional decoder), this is all that needs to be done to convert soft M-ary metrics to soft binary metrics.

When conventional single-metric decoding is employed, the difference between the two competing M-ary metrics must be converted to a single binary symbol metric that the decoder can use in the same manner as if it had been produced by a PSK demodulator. This conversion uses two design parameters: a quantizer scale factor. and a quantizer exponent.

The quantizer scale factor is used to convert the difference between competing M-ary metrics to a single binary symbol metric for conventional soft-decision decoding. The scale factor can strongly affect decoder performance and must be optimized experimentally. The allowable range for this parameter is from one to 255. It is suggested that the user experiment to find an optimum scale factor, starting with a value around twice the demodulator design-point noise quanta and then trying higher and lower values. Alternatively, this process can be avoided altogether by using the dual-metric decoding option, which is highly recommended.

The quantizer exponent provides the option of using nonlinear scaling in forming the single binary symbol metric. The allowable range for this parameter is zero to 3.5. A value of zero (or unity) provides linear quantization. Dual metric operation does not involve this parameter.

The binary to M-ary converter is implemented in run-time routine B2MARY, with initialization routine IB2MRY. The M-ary to binary converter is implemented in run-time routine MARY2B, with initialization routine IMRY2B.

4.11 BINARY CONVOLUTIONAL CODING.

The use of forward error-correction coding is vitally important to achieve robust communications in the presence of noise, interference, and other signal disturbances. In fading channel applications, the primary objective of coding is to provide acceptable end-to-end performance while allowing the demodulator to operate at quite high average error rates. Convolutional codes, when used in conjunction with interleaving or other forms of diversity, are well suited to this purpose and are among the most popular types of error-correction codes for fading channel applications. Many such codes exist; the class of rate 1/N binary codes is considered in this section.

4.11.1 Convolutional Encoder.

A convolutional encoder is a simple device consisting of a shift register and two or more modulo-two adders. For the class of rate 1/N binary codes, the number of shift register stages, K, determines the code constraint length, which is a measure of code strength. The number of modulo-two adders, N, determines the code rate. A rate 1/N code is one in which N output code bits are produced for each input data bit. The specific code is determined by the manner in which the modulo-two adders are connected to the shift register stages. The connection patterns are referred to as code generators and are traditionally given in octal format.

An example of a convolutional encoder is shown in Figure 4-7. This encoder produces a popular constraint length 7, rate 1/2 code. The octal code generators in this example are seen to be 133, 171. Data bits are shifted into the register one bit at a time. For each input bit, two code bits are provided as outputs in this example.

COMLNK enables a wide variety of rate 1/N codes to be specified. The number of modulo-two adders, which is the number of output code bits produced for each input data bit, can be specified from 2 to 30. This provides code rates from 1/2 to 1/30. While rate 1/2 codes are commonly encountered, code rates from 1/3 to 1/8 provide much better performance in Rayleigh fading channels (Bogusch, 1989b). Even lower code rates may be required to mitigate the effects of loss of phase coherence in low data rate links operating in fast fading.

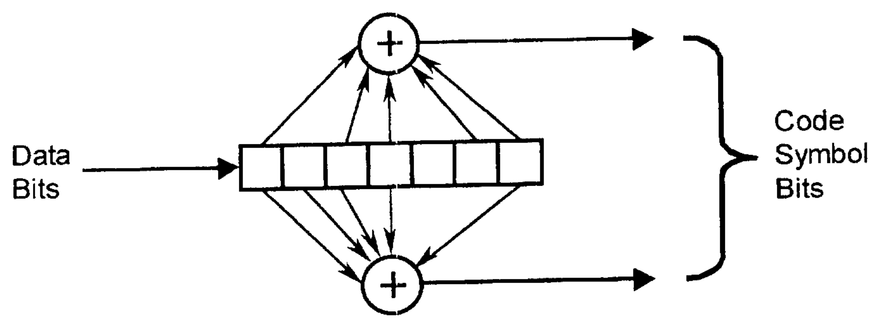


Figure 4-7. Rate 1/2, K=7 convolutional encoder.

For each code rate, the constraint length (the encoder register length) can be specified in the range from three to 28. The decoder computational requirements and memory increase exponentially with the constraint length. Constraint length seven codes (such as that depicted above) are commonly encountered, but longer constraint lengths provide more powerful codes at the expense of increased computation in the decoder. Constraint lengths in the range from seven to around 14 are practical and merit consideration.

The code generators (the bit patterns that determine the modulo-two adder tap connections) are also user-specifiable design parameters. Thus, any code can be specified. Most good codes have the property that each generator is connected to several shift register stages, including the first and last.

If each of the modulo-two adders is connected to an odd number of encoder shift register stages, the code is said to be transparent. A transparent code is one for which an inverted set of code bits is a valid code sequence corresponding to an inverted sequence of data bits. This property is desirable in coherent PSK links when phase ambiguities are resolved after error-correction decoding. The code produced by the example shown above is transparent. This property is of no consequence in noncoherent FSK and differentially coherent PSK links.

The code generators can be viewed as Galois Field 2 polynomials. If these generator polynomials have a common factor, the code is said to be catastrophic. A catastrophic code is one for which the decoder can enter an erroneous state and fail to emerge. An unbounded number of decoded bit errors can therefore result with a finite number of demodulation errors. The probability of such a catastrophic failure increases with decreasing signal-to-noise ratio (increasing demodulation error rate). It is especially likely to occur in fading channels. COMLNK automatically checks for catastrophic codes and displays a warning to the user if one is specified.

A set of known good codes is included in the program and can be accessed by entering zero for the first generator. For constraint length 14 or less, this provides one of the best known codes (others may be just as good). For $K \geq 15$ the user should specify his own code as the one supplied by the program is not necessarily very good. The built-in codes through constraint length 14 are listed in Table 4-3. These codes are primarily obtained from (Lee, 1985) and (Lee, 1986). The codes for $K = 14$ are from (Larsen, 1973).

Lower-rate codes can be formed from combinations of those listed below. For example, a rate 1/9 code can be formed by combining the code generators for a rate 1/2 code with those for a rate 1/7 code. A rate 1/30 code can be formed by combining the generators for rates 1/2, 1/3, 1/4, 1/5, 1/6, and 1/7, with the rate 1/3 generators used twice. Many other possible code rate combinations can also be constructed from the codes listed in Table 4-3.

The binary convolutional encoder is implemented in run-time routine NCODEB. The state variable dataset of this run-time module is initialized by a separate routine, INCODB.

Table 4-3. Known good convolutional codes through constraint length 14.

Code Rate	Constraint Length, K	Octal Generators			
1/2	3	5	7		
	4	15	17		
	5	23	35		
	6	45	77		
	7	133	171		
	8	247	371		
	9	561	753		
	10	1151	1753		
	11	2671	3643		
	12	4627	6765		
	13	12767	16461		
	14	21675	27123		
1/3	3	5	7	7	
	4	13	15	17	
	5	25	33	37	
	6	47	53	75	
	7	133	145	171	
	8	233	251	357	
	9	471	537	665	
	10	1117	1261	1735	
	11	2373	2671	3165	
	12	4665	5611	7473	
	13	11565	12277	16331	
	14	21645	35661	37133	
1/4	3	5	5	7	7
	4	11	13	15	17
	5	23	25	35	37
	6	45	55	73	77
	7	113	127	155	171
	8	231	277	335	353
	9	473	513	671	765
	10	1151	1345	1547	1753
	11	2351	2755	3077	3453
	12	4627	5221	6733	7635
	13	11165	13277	14331	17227
	14	21113	23175	35527	35537

Table 4-3. Known good convolutional codes through constraint length 14 (Continued).

Code Rate	Constraint Length, K	Octal Generators						
1/5	3	5	5	7	7	7		
	4	11	13	15	15	17		
	5	25	27	31	35	37		
	6	45	55	71	73	77		
	7	113	127	155	171	175		
	8	235	251	263	337	367		
	9	471	537	543	665	751		
	10	1135	1217	1323	1537	1731		
	11	2265	2363	2575	3161	3673		
	12	4317	5513	5723	6711	7725		
	13	10565	11275	12337	14471	17633		
	14	21645	21675	27123	35661	37133		
1/6	3	5	5	5	7	7	7	
	4	11	13	15	15	17	17	
	5	23	25	27	33	35	37	
	6	45	51	55	67	73	77	
	7	113	123	127	155	171	175	
	8	237	257	265	331	351	363	
	9	467	513	571	663	721	765	
	10	1067	1157	1371	1453	1651	1755	
	11	2173	2277	2423	2671	3165	3275	
	12	4317	4553	5161	5723	6671	7725	
	13	10473	11275	12467	13277	14331	16365	
	14	21113	21675	23175	27123	35527	35537	
1/7	3	5	5	5	7	7	7	7
	4	11	13	13	15	15	17	17
	5	23	25	27	31	33	35	37
	6	45	51	55	63	67	73	77
	7	113	117	123	127	155	171	175
	8	231	251	277	335	345	353	373
	9	427	475	531	551	665	737	763
	10	1123	1261	1337	1535	1651	1731	1747
	11	2167	2375	2457	2733	3151	3261	3625
	12	4317	4713	5337	5723	6711	7325	7721
	13	10473	11275	12467	13277	14331	16365	17661
	14	21113	21645	23175	35527	35537	35661	37133

Table 4-3. Known good convolutional codes through constraint length 14 (Continued).

Code Rate	Constraint Length, K	Octal Generators							
1/8	3	5	5	5	7	7	7	7	7
	4	11	13	13	15	15	15	17	17
	5	23	25	27	31	33	35	35	37
	6	45	51	55	57	63	67	73	77
	7	113	117	127	133	151	165	171	175
	8	225	237	267	275	323	331	353	371
	9	453	461	551	571	637	665	735	767
	10	1123	1165	1277	1327	1433	1575	1621	1731
	11	2155	2265	2361	2763	3167	3375	3453	3651
	12	4317	4713	5337	5723	6545	6711	7325	7721
	13	10473	11261	12475	13277	14331	15345	16751	17467
	14	21113	21675	23175	27123	35527	35537	35661	37133

4.11.2 Viterbi Decoder.

Maximum-likelihood decoding of convolutional codes is performed using the Viterbi algorithm. This algorithm can use either hard binary decisions or soft symbol metrics from the demodulator. Soft decision decoding, wherein the demodulated symbols are quantized to more than two levels to convey a measure of quality, provides substantial performance improvement in Rayleigh fading.

The choice of hard or soft decision decoding is specified by the user in terms of the maximum quantization level. This value is the largest symbol metric that will be provided as input to the decoder. Symbol metrics have values that range from zero to this maximum level.

A maximum quantization level of one yields hard decision decoding. A maximum level of seven corresponds to the popular choice of 3-bit soft decisions, while a value of 15 gives 4-bit soft decisions. Any value up to 255 (up to 8-bit soft decisions) can be specified.

The maximum quantization level is not restricted to the set of values 2^Q-1, where Q is the number of quantization bits. For example, an even value can be specified to give an odd number of quantization levels; this produces a central null or "erasure" zone.

The decoder path memory bit length is also a user-specifiable design parameter. This is the number of path history bits retained in the decoder for each code state prior to making a decision as to which path is most likely. The path memory must not be less than the code constraint length. Values around four to five times the constraint length are generally recommended. However, due to the constraints imposed by the word length of a 32-bit processor, the decoder path memory is limited to 32 bits in COMLNK. For the same reason, a 32-bit decoder path memory is often found in current equipment.

While more complex than the encoder, the Viterbi maximum-likelihood decoder is an elegant and remarkably simple machine. Even though the Viterbi algorithm is elegantly simple, it may impose significant computational requirements because of the large number of code states that must be treated with large constraint lengths.

The COMLNK implementation of the Viterbi decoder incorporates an innovation that merits discussion, *viz.*, there can be one or two input symbol metrics. The original application of the

Viterbi algorithm involved BPSK links, for which the demodulator produces a single bipolar decision variable. For such links, it is reasonable to provide the decoder with a single symbol metric since the competing metric can be inferred directly. Thus most, if not all, implementations of the Viterbi algorithm currently found in equipment accept only a single symbol metric as input.

A single metric is not well suited to FSK links, which do not produce a bipolar decision variable. Indeed, one of the problems that a designer encounters when interfacing a Viterbi decoder with an M-ary FSK demodulator is how to form a single decision variable from M matched filter outputs. The M-ary-to-binary converter discussed earlier contains an algorithm for accomplishing this. While the algorithm is commonly used, it is not optimal.

Here, we introduce the concept of dual-metric decoding in FSK links. The concept is described most easily by first considering binary FSK. In a matched-filter implementation, a binary FSK demodulator forms two filters centered at the two possible transmitted frequencies. The signal plus noise in each filter is integrated over the symbol period, and the filter with the larger output is considered to contain the signal. With hard decisions, there is nothing more to consider. With soft decisions, however, we note that the magnitude of each filter output provides a measure of the likelihood of that filter containing the signal. If one output far exceeds the other, high confidence should be assigned; if the two outputs are approximately equal, there is low confidence. The maximum amount of information can therefore be given to the decoder simply by providing both filter outputs, one for each of the two binary values.

Extension of this concept to M-ary FSK is straightforward. For each bit position in an M-ary symbol, exactly M/2 filters correspond to a zero in that position; the other M/2 filters correspond to a one. Find the maximum value of each of these two sets of filters and provide these values to the decoder as the two competing symbol metrics. This algorithm is included as an option in the COMLNK M-ary-to-binary converter routine.

Examination of the COMLNK implementation of the binary Viterbi decoder reveals that the dual-metric design does not require more computation than a conventional single-metric design. In fact, it requires slightly less computation. The explanation is that the first operation in a conventional decoder is to compute the competing metric from the one given, assuming a bipolar decision variable. While the difference in computational requirements is trivial, the point is that the use of dual metrics provides better performance at no cost.

The binary Viterbi decoder is implemented in run-time routine DCODEB. The state variable dataset for this run-time module is initialized by a separate routine, IDCODB.

4.12 M-ARY CONVOLUTIONAL CODING.

Several different types of nonbinary codes can be considered, including dual-k and triple-k convolutional codes. These two codes, along with one or more Reed-Solomon block codes, are planned for future inclusion in COMLNK. At present, one specific type of M-ary convolutional code is available, as described below.

An extension of the concept of dual-metric decoding (see the preceding discussion of binary convolutional coding) leads to the realization that M-ary FSK modulation is precisely matched to rate 1/N convolutional encoding when $N = \log_2 M$. As noted by (Bogusch, 1989b), when this relationship is satisfied each M-ary symbol can contain the set of binary code symbols that correspond exactly to one branch in the code tree. The M matched filter outputs of the FSK demodulator then directly provide the M branch metrics needed in the Viterbi decoding algorithm. For example, a rate 1/2 code is matched to 4-ary FSK, a rate 1/3 code is matched to 8-ary FSK, a rate 1/4 code is matched to 16-ary FSK, and so on.

Examination of the binary decoder reveals that it consists of two principal sections: (1) a section in which binary symbol metrics are input and converted to code branch metrics, and (2) the actual decoding section in which the most likely path entering each code state is determined. The first section just accepts the number of binary symbol metrics that correspond to one code branch (for example, three for a rate 1/3 code), and adds one of the two competing binary metrics to each branch metric (eight for a rate 1/3 code). Which of the two binary metrics is added to a given branch metric depends on whether the branch contains a zero or a one in that bit position. What the decoder is actually doing amounts to binary-to-M-ary reconstruction of a set of matched filter outputs to determine the likelihood of each branch!

Given this insight, it is clear that it would be much better to simply provide the demodulator matched filter outputs to the decoder directly, without M-ary-to-binary conversion and subsequent binary-to-M-ary reconstruction. This can be easily done when the inverse code rate, N, is matched to the M-ary modulation alphabet as defined above, *i. e.*, when $M = 2^N$.

Thus, when M-ary FSK modulation is employed, the user has the option of selecting a rate $1/\log_2 M$ convolutional code to be used as an M-ary code. The corresponding decoder is identical to the binary Viterbi decoder, except that the first section to compute the branch metric is no longer needed and hence is discarded. This simplifies the decoder while providing an optimum set of soft-decision metrics.

The advantage gained by the improved metrics must be weighed against the potential disadvantage of having all of the code symbol bits contained in a single modulation symbol that may be lost in a signal fade. The ability to perform such design trades in a wide range of signal conditions is COMLNK's *raison d'être*.

The binary-to-M-ary convolutional encoder is implemented in run-time routine NCODEM, whose state variable dataset is initialized by routine INCODB. The M-ary-to-binary Viterbi decoder is implemented in run-time routine DCODEM, whose state variable dataset is initialized by routine IDCODB.

4.13 SYMBOL REPETITION.

While inelegant, simple repetition of transmitted symbols with corresponding accumulation at the receiver is one of the oldest types of error-correction code. It is often used in conjunction with more powerful codes to achieve a lower overall code rate. This is useful to reduce the symbol period in low data rate links that must operate in fast fading channels.

Repetition can be performed at various points within a link, giving rise to such terms as bit repetition (when done at the binary level) and chip repetition (when done on M-ary symbols prior to transmission). In fading channel applications, repetition (and coding in general) is most effective when followed by interleaving. If repetition is done immediately after convolutional encoding, one should compare the resulting performance with that of an equivalent lower-rate convolutional code.

In COMLNK, repetition can be inserted anywhere in a link layout, as often as desired. The repetition factor is the number of times that each input symbol is transmitted. A factor of one implies no repeat while a factor of two causes each symbol to be sent twice. The allowable range for this factor is one to 65535 ($2^{16} - 1$). The associated combining operation in the receiver chain accumulates either hard or soft decision metrics, depending on the operations preceding the combiner.

The symbol repeater is implemented in run-time routine REPEAT, with initialization routine IREPET. The symbol combiner is implemented in run-time routine COMBIN with initialization routine ICOMBN.

4.14 BLOCK INTERLEAVING.

Successful communications in fading channels depends on the combination of error-correction coding with some type of diversity. Diversity is required to convert the burst-error channel into something resembling a random-error channel, for which convolutional codes are suited. Even so-called burst-error correction codes generally require the use of diversity techniques in fading radio channels because error bursts are often orders of magnitude longer than any practical code can tolerate.

Interleaving is an easily implemented form of time diversity and is particularly well suited to low or medium data rate links for which memory and delay requirements are modest. Interleaving the encoded symbols prior to transmission and deinterleaving them prior to decoding causes demodulation errors to be scrambled, producing a more nearly random-appearing error pattern to the decoder. If the interleaving span is much longer than the longest error burst, essentially the full random-error correction capability is restored to a Viterbi decoder.

Several methods of implementing interleavers are found in practice. Perhaps the most straightforward implementation is a block interleaver, in which encoded symbols are written into a two-dimensional array in one order and then read out of the array in a different order. To achieve synchronous operation, two identical arrays are required at the transmitter, with one being written while the other is being read. A similar double-buffered configuration is required at the receiver. Thus, the total delay is twice that of a single array, and the total memory is at least four times larger (soft-decision decoding in an FSK modem further increases the amount of deinterleaver memory). Although simple to implement, block interleaving requires the most memory and imposes the greatest delay for a given interleaving span of any technique.

A block interleaver can be described as a row-column array, in which encoded symbols to be transmitted are written into columns of the array. When the array is filled, the symbols are read out of rows. The number of rows, n_1, determines the separation that two adjacent demodulation errors will have at the decoder input after deinterleaving and is thus related to decoder memory. The number of columns, n_2, determines the separation of adjacent encoded input symbols in the channel and hence is related to channel memory or fade duration.[1]

With convolutional codes, it is desirable to separate correlated errors by several code constraint lengths. Thus, the value of n_1 should be on the order of 3KN, where K is the code constraint length and N is the inverse code rate. The value of n_2 should then be chosen so that the resulting span, n_1n_2, is at least $30R_s\tau_{o\max}$, where $\tau_{o\max}$ is the largest scintillation decorrelation time specified for the link, and R_s is the rate at which symbols are written into the interleaver (Bogusch, 1989b).

In COMLNK, any number of rows and columns can be specified, subject only to the amount of memory available. Interleaving can be either fixed or pseudorandom. Pseudorandom interleaving involves permuting the order of the rows that are read out of the interleaver, with a different order chosen for each block.

Block repetition can also be specified, whereupon each interleaver block is transmitted a specified number of times. The block repeat factor is the number of times that the interleaver array is read out and transmitted prior to filling it with the next set of encoded symbols. This method of

[1] In some applications, the block interleaver is filled by writing into rows and emptied by reading out of columns. The choice is arbitrary as long as the same convention is applied at both ends of the link and the dimensions are properly chosen. Simply interchange the words "row" and "column" in this description to obtain the alternate convention.

repetition provides maximum separation (the interleaver size) between repeated symbols, and can provide a significant advantage over conventional symbol repetition in slow fading channels.

At the receiver, the deinterleaver operates in a manner that restores the original order of the encoded symbols prior to feeding them into a decoder. Thus, the deinterleaver writes into rows and reads out of columns. If pseudorandom interleaving is specified, the row write order is permuted using the same algorithm as used in the interleaver. If block repetition is specified, symbol accumulation is performed in the deinterleaver. When M-ary FSK demodulation is employed, the deinterleaver must store and deinterleave the entire set of M matched filter outputs for each symbol.

The block interleaver is implemented in two run-time routines, ILBLRI and ILBLKI, depending on whether block repetition is or is not specified, respectively. The corresponding run-time deinterleaver modules are ILBLRD and ILBLKD, respectively. For either pair of modules, initialization of the interleaver state variable dataset is performed by routine IILBLI, while the deinterleaver state variable dataset is initialized by routine IILBLD.

4.15 CONVOLUTIONAL INTERLEAVING.

A convolutional interleaver enables synchronous interleaving to be performed with one-half the delay and about one-fourth the memory of a block interleaver. In effect, the convolutional interleaver takes one row-column array, slices it diagonally, and places one half at the transmitter and the other half at the receiver. The portion at the receiver is, of course, the deinterleaver. Precise comparisons of memory requirements between the two types of interleavers must take into account the fact that in any implementation there must be sufficient storage in the deinterleaver to accommodate soft decisions. In an M-ary FSK system, this requires M bytes in the deinterleaver for each byte in the interleaver.

The convolutional interleaver can be pictured as a set of shift registers of different lengths, usually tapered from a maximum length down toward zero length, or vice-versa. The number of registers is analogous to the number of rows, n_1, in a block interleaver. The length of the largest register, n_2, is analogous to the number of columns. The span is still defined as the product n_1n_2.

Hence, the number of columns is the longest row length, which is related to channel memory or decorrelation time, while the number of rows is related to code memory (constraint length and code rate). The design guidelines given for choosing these parameter values in the block interleaver apply here as well.

Many variations of the convolutional interleaver are possible. Not only can the values of n_1 and n_2 be chosen at will, the register lengths need not vary linearly or even monotonically. However, linear monotonic variation is the design configuration employed in COMLNK.

The implementation of the convolutional interleaver employs an exceptionally simple algorithm devised by (Newman, 1983). The algorithm uses a circular storage array with a read pointer and a write pointer for each row (or register). The pointers advance around the circular array in an inchworm fashion. As with the block interleaver, interleaving can be fixed or pseudorandom. Pseudorandom interleaving involves permuting the row read order, with a different order chosen at the end of each cycle through rows.

The convolutional deinterleaver restores the original order of the encoded symbols prior to feeding them into the decoder. The deinterleaver is the same device as the interleaver except that the tapered rows are arranged in the opposite order. Therefore, the combined length of the corresponding interleaver and deinterleaver rows is the same for each pair and equals the total delay through the two devices.

The deinterleaver implementation utilizes the same type of circular storage array with sets of row write and read pointers. If pseudorandom interleaving is performed, the deinterleaver row write order is permuted using the same algorithm as in the interleaver. In any case, the deinterleaver stores and deinterleaves the entire set of matched filter outputs when M-ary FSK demodulation is employed.

The convolutional interleaver is implemented in run-time routine ILCNVI. The state variable dataset of this run-time module is initialized by routine IILCNI. The convolutional deinterleaver is implemented in run-time routine ILCNVD. The state variable dataset of this run-time module is initialized by routine IILCND.

4.16 TIME-DIVISION MULTIPLEXING.

Time-division multiplexing (TDM) is incorporated in COMLNK to provide accurate simulation of links wherein data from multiple sources are interspersed in a single digital stream. The data for a single user is processed prior to the multiplexer and after the demultiplexer. In between, it is necessary to process data from all sources to achieve proper timing relationships. For example, multiplexing affects the timing of signal fading or pulse jamming relative to the demodulation of channel symbols corresponding to a given user. Tracking loop operation is also affected by multiplexing, as are all functions that occur in the multiplexed portion of the link.

Another application of multiplexing involves stuffing extra bits into a data stream to achieve a desired numerology such as frame size or transmission rate. Extra bits may also be inserted for purposes of synchronization or encoder flushing, for example. Note, however, that such applications can also be handled by the sync multiplexing module discussed in the following subsection.

The distinction between TDM, discussed here, and sync-symbol multiplexing, discussed subsequently, involves who is charged for the energy required to transmit the additional data. With any of the TDM algorithms discussed here, the user is not charged for the energy required to transmit the additional data. Transmitter power is increased as needed to maintain the symbol energy.

Whatever the application, many different algorithms can be used to intersperse data from different sources. Three options are considered here. These are uniform multiplexing and two forms of nonuniform multiplexing wherein the extra symbols are grouped together at the front or the rear of a TDM frame.

4.16.1 Uniform Multiplexing.

Uniform interspersing of user and other data is a simple and common choice. The specific algorithm adopted here is chosen for ease of implementation. The interrupt-driven architecture in COMLNK allows a module to place its output on the data bus at any time. The next module can access the bus at any subsequent time. The rate at which data for a module is placed on the bus, and the rate at which another module accesses that data, can therefore be quite different and the two rates do not have to exhibit a simple relationship. Bus access is continually monitored by the interrupt controller, which transfers control to the module with the earliest pending interrupt. The module then returns control to the interrupt controller, and the process continually repeats.

With this architecture, uniform multiplexing is accomplished simply by placing data from a given user on the bus at the single user rate, with the multiplexer accessing the bus at the multiple user rate. The demultiplexer performs the reverse operation, putting multiplexed data on the bus at the multiple user rate, which is then accessed by the following module at the single user rate. The result is that data is processed at the multiple user rate between the two TDM modules, while the data from a single user flows into and out of the two modules.

The resulting multiplexed data are uniformly interspersed in time. For example, if user data constitutes 3/17 of the total, three out of every seventeen symbols transmitted belong to the user of interest, and these three user symbols are approximately evenly spaced in each seventeen-symbol frame. Only user data is processed through the modules that precede the multiplexer and follow the demultiplexer. The other data, which is undefined, is multiplexed to achieve proper timing but discarded later at the demultiplexer.

4.16.2 Nonuniform Multiplexing.

Two forms of nonuniform interspersing of user and other data are available. The other data can be grouped together at either the beginning or the end of the TDM frame. This requires that data be buffered to provide the specified TDM frame format. Synchronous operation is achieved by using double buffers in both the multiplexer and demultiplexer modules. These buffers are implemented in 8-bit memory (byte arrays) to reduce storage requirements.

With either uniform or nonuniform multiplexing, one or more user symbols and one or more other symbols can be specified in the TDM frame. An option is provided to either reduce the input symbol rate or increase the output symbol rate of the multiplexer to accommodate the multiplexed data.

Examination of the modules that implement the uniform TDM process reveals that it is much simpler to implement the process in the code than it is to describe it in the text. Nonuniform TDM is only marginally more complicated because of the buffering requirements.

In either case, the most complex part of the implementation is handled by interrupt timing of the module that follows the demultiplexer. Pending interrupts of that module must be delayed as much as possible so that a demultiplexed symbol is taken off the data bus just before the next symbol is placed on the bus. This ensures that the user symbol stream is properly demultiplexed for all possible channel allocations. Because any number of user symbols can be multiplexed with any number of other symbols, resulting timing relationships can become quite complex. The user need not be concerned with such details, however, as they are handled automatically by the program.

The uniform multiplexer is implemented in run-time routine MUXU, and the nonuniform frame-buffered multiplexer is implemented in routine MUXF. The corresponding demultiplexers are implemented in routines DEMUXU and DEMUXF, respectively. The multiplexer modules are initialized by routine IMUX, and the demultiplexers are initialized by routine IDEMUX. Multiplexing energy costs are allocated by routine ANALYZ, and interrupt timing for all modules is initialized by routine LAYOUT.

4.17 SYNC-SYMBOL MULTIPLEXING.

Some FSK systems insert periods of unmodulated carrier, commonly referred to as sync symbols, into the transmitted data stream to facilitate carrier acquisition and tracking at the receiver. This time-division multiplexing operation does not necessarily produce uniform interspersing of sync and data symbols. Instead, the process involves a synchronization frame that contains specified numbers of sync and data symbols, which may be multiplexed in a pseudorandom or other nonuniform manner.

Other applications may involve insertion of extra bits or symbols into a data stream to achieve a desired frame size or transmission rate. Extra bits also may be inserted for encoder flushing, for example.. Note, however, that such multiplexing operations can also be handled by the TDM module discussed in the preceding subsection.

The distinction between sync multiplexing, discussed here, and TDM, discussed previously, involves who is charged for the energy required to transmit the additional symbols. With any of the sync multiplexing algorithms discussed here, the user is charged for the energy required to transmit the extra symbols. Therefore, transmitter power is not increased, and an increased transmission rate leads to a decreased energy per symbol.

Whatever the application, many different algorithms can be used to multiplex sync or other symbols with data. Four options are considered here. These are uniform multiplexing, pseudorandom multiplexing, and two forms of nonrandom nonuniform multiplexing wherein the extra symbols are grouped together at the front or the rear of the transmission frame.

4.17.1 Uniform Multiplexing.

Uniform multiplexing of sync and user data is accomplished in exactly the same manner as described in the preceding TDM discussion. The algorithm makes use of the interrupt-driven architecture in COMLNK to uniformly intersperse user data with sync symbols. No buffer is required in this case. Refer to the preceding subsection for discussion of uniform multiplexing.

4.17.2 Pseudorandom Multiplexing.

Pseudorandom multiplexing involves permutation of the sync frames prior to transmission. In this case, two buffers are required in the multiplexer for synchronous operation, similar to a block interleaver. Data and sync symbols are written into one buffer while the other is being read out. When the first buffer is filled and the second is empty, the two buffers are swapped by exchanging pointers, and the process continues.

One or more data symbols can be specified in each sync frame, along with zero or more sync symbols (zero sync allows data frame permutation only; see the next section). The resulting frame is pseudorandomly permuted prior to being read out, resulting in a scrambled transmission sequence. Each sync frame is independently permuted before transmission.

At the receiver, the pseudorandom demultiplexer performs the complementary operation of stripping out the sync symbols and restoring the original order to the data sequence. The demultiplexer also uses double buffering for synchronous operation. All buffers are implemented in 8-bit memory (byte arrays) to reduce storage requirements.

4.17.3 Nonuniform Multiplexing.

Two forms of fixed nonuniform multiplexing are available. Sync symbols can be grouped together at either the beginning or the end of the sync frames. This again requires double buffering and is accomplished in exactly the same manner as described in the preceding TDM subsection.

With any of the multiplexing options, the user can specify whether the input rate is to decrease or the output rate is to increase to accommodate the sync symbols. Note again that with this module an increased symbol transmission rate results in less energy per symbol.

The uniform sync multiplexer is implemented in run-time routine MUXU. The pseudorandom multiplexer is implemented in routine MUXP, and the fixed nonuniform multiplexer is implemented in MUXF. The corresponding demultiplexers are implemented in routines DEMUXU, DEMUXP, and DEMUXF, respectively. The multiplexer modules are initialized by routine IMUX, and the demultiplexers are initialized by routine IDEMUX. Multiplexing energy costs are allocated by routine ANALYZ, and interrupt timing for all modules is initialized by routine LAYOUT.

4.18 TIME PERMUTATION.

Time permutation is a technique that is sometimes used as part of an overall strategy to mitigate the effects of jamming or other periodic interference. This function can be implemented by partitioning the transmitted symbol sequence into frames, with each frame pseudorandomly reordered prior to transmission. The inverse function is performed at the receiver to restore the original symbol order.

A sample without replacement permutation algorithm is used to perform the pseudorandom reordering operation. This algorithm is implemented in routine SCRMBL. The initial random number generator seed for the permutation is a user input.

When implemented with double buffering for synchronous operation, this time permutation function becomes the same as the preceding sync-symbol multiplexer when it is operating in a pseudorandom manner, except that here no extra symbols are multiplexed into the data frame. Thus, the same pair of modules as used for sync-symbol pseudorandom multiplexing are used to perform the time permutation function, simply by setting the number of sync symbols to zero.

Frame permutation is implemented in run-time routine MUXP, with initialization routine IMUX. The depermuter is implemented in run-time routine DEMUXP, with initialization routine IDEMUX.

4.19 FSK FRAME ACQUISITION USING SOFT M-ARY PREAMBLE CORRELATION.

Frame synchronization may be required to identify the beginning of a message or data block. This requirement becomes particularly important in systems that involve asynchronous communications, wherein transmission may begin at any time with a variable amount of dead time between messages. The module described in this section is designed to provide signal acquisition and frame synchronization in asynchronous FSK systems that use unhopped (or slowly hopped) waveforms.

4.19.1 Assumptions.

The signal waveform employs M-ary FSK modulation. Each transmission frame begins with a sequence of pseudorandom symbols referred to as the preamble. During the time required to transmit the preamble, the carrier frequency is known to the receiver. The preamble length and symbol sequence are also known to the receiver. The timing of each transmission is uncertain over a range of plus or minus K symbol periods. A guard band is inserted at the beginning and end of each frame to allow for this uncertainty. Hence, each transmission frame has the format depicted in Figure 4-8.

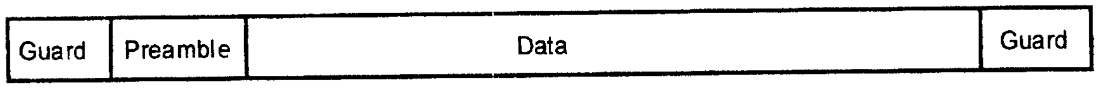


Figure 4-8. Transmission frame format.

4.19.2 Acquisition Procedure.

The receiver begins to demodulate the incoming waveform at the expected time of signal arrival. Demodulation is performed using standard M-ary FSK matched filter detection, providing M matched filter outputs at the end of each symbol period. The matched filter outputs are digital values, quantized to any desired number of bits. In this software, 8-bit bytes are used for filter outputs, and 32-bit words are used for accumulated values. Denote the matched filter outputs at the i-th symbol time as

$$F_i(m) \quad , \qquad m = 0, 1, 2, \ldots, M-1$$

As each symbol is demodulated, the matched filter outputs are correlated with the known symbol sequence, advanced and retarded over the K-symbol uncertainty range. Thus a total of 2K+1 correlations are formed. Each correlation is an accumulation of the output of the matched filter corresponding to the symbol that would be received at that delay. Denote the known preamble sequence as *m(i+k)* , where *i* is the on-time symbol index, and *k* is an advance or delay relative to the on-time symbol:

$i = 0, 1, 2, \ldots$	on-time symbol index
$k = -K, \ldots, -2, -1, 0, 1, 2, \ldots, K$	delay bin index

At any time in the acquisition process, the correlation accumulations may be written as follows. This technique provides a maximum-likelihood estimate of the correlation between the received signal and each of the 2K+1 delay hypotheses, while requiring only 2K+1 additions to be performed each symbol period:

$$\begin{aligned} C_n(k) &= \sum_{i=0}^{n} F_i[m(i+k)] \\ &= C_{n-1}(k) + F_n[m(n+k)] \qquad , \qquad k = -K, \ldots, +K \end{aligned}$$

Consider for the moment a case in which the actual timing of the received signal is precisely aligned with one of the accumulation delay bins. Then each matched filter output in this accumulation contains the signal, plus noise. If the symbol in a different delay bin happens to be the same as the correct symbol, then signal plus noise will be added to that accumulation as well. With a pseudorandom symbol sequence, this will occur with probability 1/M. When the symbol at a different delay does not correspond to the actual symbol, only noise will be added to the accumulation. This occurs with probability (M-1)/M.

Let *S* denote the average value of the signal and *N* denote the average value of the noise in the matched filter outputs. Then the maximum accumulation, corresponding to the actual signal delay, may be approximated by

$$C_{max} \approx n(S+N)$$

where *n* is the number of symbols included in the summation. The other accumulations may be approximated by

$$C_{av} \approx n(S/M+N)$$

where the notation C_{av} implies an average over all except the maximum accumulation. Thus, by finding the maximum value and the average of the other values, an estimate of the signal-to-noise ratio is given by

$$S/N \approx \frac{C_{max} - C_{av}}{C_{av} - C_{max}/M}$$

This SNR estimate can be compared to a threshold to decide whether the signal has been acquired. If so, the delay bin index of the maximum accumulation provides the adjustment needed to align the receiver timing with the actual signal timing.

Now consider the general case in which the signal symbol timing is not precisely aligned with one of the delay bins, but instead straddles two of the receiver symbol periods. Let τ denote the magnitude of the difference between the time of symbol transitions in the received signal and that in the receiver. The value of τ is zero when the signal is precisely aligned with one of the delay bins. The value of τ is one-half when the symbol transitions in the received signal fall in the middle of the receiver symbol period. This is the worst case, because further displacement of the signal timing simply causes it to align more closely with an adjacent delay bin.

The foregoing approximation for the maximum correlation accumulation may be rewritten in terms of the fractional symbol period error τ as follows, together with a corresponding approximation for the second-largest accumulation:

$$C_{max} \approx n[(1-\tau)S + \tau S / M + N]$$
$$C_{2^{nd}} \approx n[\tau S + (1-\tau)S / M + N]$$

It can be shown that an estimate of the fractional error in symbol timing is given by

$$\tau \approx \frac{C_{2^{nd}} - C_{av}}{C_{max} + C_{2^{nd}} - 2C_{av}}$$

where C_{av} is now the average of all but the two largest accumulations.

4.19.3 Summary.

The acquisition algorithm can be summarized as follows. The 2K+1 correlations are accumulated over a period from the expected start of the signal to the time that the preamble would end if the signal had the least anticipated delay. This corresponds to the known length of the preamble minus K symbols. The value of K is known because it is the largest uncertainty that the receiver is designed to acquire.

Once the accumulations are formed, the receiver determines the largest value, second largest value, and the average value of those remaining. An estimate of the signal-to-noise ratio is obtained using the largest and average values. If this estimate exceeds a threshold level (a design value), acquisition is deemed successful.

Upon successful acquisition, receiver timing is given a discrete shift based on the delay bin indexes of the maximum and second maximum correlations. If these two values reside in adjacent delay bins, the fractional symbol period error τ is estimated and used to adjust the receiver timing between the delays of the two largest values. In the event that the second largest correlation is not in a delay bin adjacent to the maximum value, only the delay of the maximum correlation is used to adjust receiver timing.

This technique not only offers rapid coarse acquisition of signal timing, it provides much of the fine timing correction as well. Note in particular that the estimate of the fractional delay error τ eliminates the potential problem of delay-lock loop hang-up that is commonly encountered when receiver timing straddles signal symbol transitions.

Residual fine timing acquisition and subsequent time tracking are typically carried out using a second-order delay-lock loop that operates with an early-late discriminator algorithm (Bogusch, 1990). This discriminator algorithm is designed for use with suppressed-carrier M-ary FSK waveforms. Therefore, time tracking is not dependent on the presence of an acquisition preamble or synchronization symbols. Indeed, in this design, delay-lock time tracking and preamble sequence acquisition are conducted simultaneously, minimizing the overall time to achieve both coarse and fine timing acquisition.

4.19.4 Design Parameters.

The number of guard band symbols K, the number of preamble symbols, and the number of data symbols per transmission frame are all user-specifiable design parameters. The output symbol rate may increase or the input rate may decrease to accommodate the guard time and preamble symbols. Receiver timing error may be fixed or random for each frame. The acquisition threshold is also an input parameter.

This acquisition algorithm is implemented in the ACQTX and ACQRX modules. The state variable datasets of these modules are set up by initialization routines IACQTX and IACQRX.

4.20 FSK FRAME ACQUISITION USING BINARY PREAMBLE CORRELATION.

Frame synchronization may be required to identify the beginning of a message or data block. This requirement becomes particularly important in systems that involve asynchronous communications, wherein transmission may begin at any time with an unknown amount of dead time between messages.

The module described in this section is designed to provide signal acquisition and frame synchronization in asynchronous systems that use unhopped binary or M-ary FSK waveforms. The fundamental differences between this module and that described in the preceding section are (1) the signal timing in this case is completely unknown, and (2) this acquisition algorithm uses hard or soft binary correlators rather than soft-decision M-ary FSK matched filter outputs.

4.20.1 Assumptions.

The signal waveform employs unhopped FSK modulation. Transmission timing is asynchronous, and the time at which a transmission begins is completely unknown to the receiver. Each transmission frame begins with a sequence of pseudorandom symbols referred to as the preamble. The preamble length and symbol sequence are known to the receiver. Following the preamble is a block of data symbols. The number of data symbols per frame is known to the receiver. There may be a postamble at the end of the frame, following the data. Contents and length of the postamble, if any, are irrelevant to the acquisition process. Hence, each transmission frame has the format depicted in Figure 4-9.

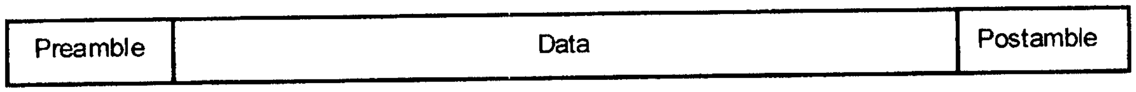


Figure 4-9. Preamble frame format.

4.20.2 Preamble Structure.

The preamble is formed by concatenating one or more pseudonoise (PN) binary sequences, each with a binary identification sequence (ID word) appended. If the basic FSK modulation format is not binary, these preamble symbols are mapped to two symbols in the M-ary alphabet prior to transmission. When more than one PN sequence is used, the sequences are identical, with only the ID word differing from one PN sequence repetition to the next. The resulting preamble structure is depicted in Figure 4-10.

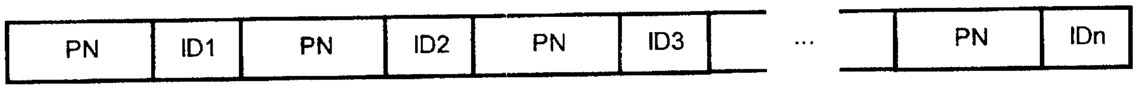


Figure 4-10. Preamble structure.

4.20.3 PN Sequence Correlation.

The receiver continually attempts to demodulate the incoming waveform, which may be only noise or signal plus noise. Demodulation is performed using standard M-ary FSK matched filter detection, providing M matched filter outputs at each sample time. Because symbol timing is unknown, several samples are formed for each symbol period. Boxcar filters are employed in the tone filter bank demodulator to produce the matched filter outputs, which are the inputs to this acquisition module at each sample time. These filter outputs are digital values, quantized to any desired number of bits. In this software, 8-bit bytes are used for the filter outputs. Denote the M filter outputs at the k-th sample time as

$$F_k(m) \ , \qquad \begin{aligned} m &= 0, 1, 2, \ldots, \text{M-1} \\ k &= 0, 1, 2, \ldots, \text{K-1} \end{aligned} \tag{4.7}$$

where M is the number of modulation levels, and K is the number of sample times per M-ary FSK symbol period (an even integer).

While searching for the start of a message, the acquisition module accepts these M values at each sample time and forms a hard binary decision. The decision is based on the outputs of two of the M filters. During preamble transmission, a binary zero is transmitted as tone M/4 and a binary one is transmitted as tone M-1-M/4, where the truncated integer value of M/4 is used if necessary. During preamble reception, the acquisition module compares the outputs of these two filters to form the binary decision:

$$\begin{aligned} D_k &= 0 \quad , \qquad F_k(\text{M/4}) \geq F_k(\text{M-1-M/4}) \\ &= 1 \quad , \qquad F_k(\text{M/4}) < F_k(\text{M-1-M/4}) \end{aligned} \tag{4.8}$$

The resulting binary value is shifted into a correlation register. There is one (L+J)-bit correlation register for each of the K sample times within a symbol period, where L is the number of bits in the PN sequence, and J is the number of bits in each ID word.

In the basic design of this algorithm, the oldest L-bit contents of each register are correlated with the known PN sequence. A correlation is simply a count of the number of agreements minus the number of disagreements. With perfect agreement, the correlation sum would be L, while with random binary values the correlation sum would be approximately zero.

When a full sample set is available, the results of all K correlations (one for each sample time in the current symbol period) are compared to a correlation magnitude threshold. If none of the correlation sums equals or exceeds the threshold, a flag is cleared to denote no correlation for this symbol period. A countdown counter R_i for the number of symbols remaining in the preamble is decremented, and the acquisition module waits for the next set of inputs from the demodulator.

If one or more of the K correlations equals or exceeds the threshold, the sample index k_a corresponding to the maximum correlation sum is determined. If two or more sample times produce identical maximum correlation sums, the average value of the corresponding set of indexes is found, along with the minimum and maximum indexes in this set (k_{min}, k_{max}). Because the indexes are integer values in the range 0 to K-1, the average value is a truncated integer:

$$k_a = \text{Int}\left(\frac{k_{min} + k_{max}}{2}\right) \tag{4.9}$$

The newest J-bit portion of the correlation register for this sample index is then compared to the set of valid ID words. This comparison is simply another correlation, in which the number of disagreements between the received ID word and each of the valid ID words is determined. The result, known as the Hamming distance, is zero for perfect agreement and approximately J/2 with random binary values.

The minimum Hamming distance is compared to a threshold to determine if a valid ID word was received at the end of the detected PN sequence. If the minimum distance is at or below the Hamming threshold, the correlation is accepted and entered into the sequence timing discussed next. Otherwise, if the Hamming distance exceeds the threshold, the detection is declared invalid. In that event, the acquisition module clears the current symbol correlation flag, decrements the countdown symbol counter, and awaits the next set of inputs.

4.20.4 PN Sequence Timing.

If the minimum distance is equal to or less than the Hamming threshold, a flag is set denoting a valid correlation for this symbol period. The index of the ID word associated with the minimum Hamming distance identifies which of the N repeated PN sequences in the preamble has been detected. Using this index, the expected number of symbols remaining in the current preamble is given by

$$R_n = (N - n)(L + J) + 1 \tag{4.10}$$

where N is the number of PN sequences in the preamble, and n is the index of the minimum distance ID word for the current PN sequence ($1 \leq n \leq N$). The additional +1 term above allows for the fact that the demodulator timing may straddle a symbol period. Thus, the best timing for the current detection may occur during either the current or the following symbol period.

If this is the first correlation for the current frame, a correlation counter is initialized to one and the countdown symbol counter, R_i, is set to the expected number of remaining symbols, R_n. A flag is set to indicate that an extra symbol count is included in the R_i counter (*i. e.*, the counter contains the +1 term shown above). The maximum correlation sum and associated average and minimum sample indexes, k_a and k_{min}, are saved. The counter R_i is then decremented, and the acquisition module is ready for the next set of inputs.

If there was a previous correlation for the current frame, the countdown symbol counter R_i, which is the expected number of remaining preamble symbols based on the previous correlation, is compared to the value R_n, which is the expected number of remaining preamble symbols based on the current correlation. For the two correlations to be consistent, the two counts should not differ by more than one symbol (allowing for the demodulator timing to possibly straddle a symbol). If this consistency check fails, the current correlation is ignored if its correlation sum is less than the previous value. Otherwise, the current correlation replaces the previous one.

If the current and previous correlations exhibit consistent symbol counts, both are valid. However, the two correlations may result from the demodulator timing straddling a symbol period and hence may actually reflect a single correlation. This is detected by examining the flag that indicates if there was a correlation on the previous symbol. If not, the current correlation is indeed a new one and the correlation counter is incremented. The remaining operations are the same as for the first correlation: the count-down symbol counter is reset to the expected number of remaining symbols, the flag indicating the presence of an extra symbol count is set, the maximum correlation sum and associated average and minimum sample indexes are saved, and the count-down counter is decremented.

4.20.5 Symbol Timing Estimation.

If there was a correlation on the previous symbol, the current correlation is an extension of the previous one and only the sample index for the best demodulator symbol timing may need to be adjusted. Three possibilities arise, depending on whether the current correlation sum is less than, greater than, or equal to the previous correlation sum

If the current correlation sum is less than that obtained on the previous symbol, the best timing for this correlation is that detected previously, and the current correlation is ignored.

If the current correlation sum is greater than that obtained on the previous symbol, the best timing for this correlation is that detected currently. In this case, the countdown counter R_i is reset to the expected number of remaining symbols R_n minus one (since the best timing is now known to occur for this symbol period), and the flag indicating the extra symbol count is now cleared.

If the current correlation sum is equal to that obtained on the previous symbol, the estimation of which sample index will provide the best demodulator symbol timing becomes more complicated. Because the correlation process uses hard binary decisions for each preamble symbol at each sample time, the possibility of ties occurring in the correlation sum is quite significant. This is especially true at relatively large signal-to-noise ratios, for which correct binary decisions are likely to be obtained over a range of timing positions around the optimum timing.

What is known in this case are the minimum sample index for the maximum correlation sum in the previous symbol period, and the maximum sample index for the same maximum correlation sum in the current symbol period. The mean sample index when correlation ties occur in consecutive symbol periods is then computed as

$$k_{avg} = \mathrm{Int}\left(\frac{k_{min1} + k_{max2} + K}{2}\right) , \qquad \text{modulo K} \qquad (4.11)$$

where k_{min1} is the minimum sample index for the tied correlation values in the preceding symbol period, k_{max2} is the maximum sample index for these tied values in the current symbol period, and K is the number of samples per symbol period.

The addition of the number of samples K in this computation accounts for the fact that the sample index during each symbol period ranges from zero to K-1, and the current symbol period is just an extension of the previous one for the purposes of this calculation. Taking the result modulo K then maps the mean sample index back into the proper range. Which symbol period the result applies to is then determined by the extra symbol flag, which is cleared if the result is in the current symbol period (k_{avg} in the range 0 to K/2-1). In addition, in this case the countdown symbol counter R_i is adjusted downward to R_n - 1.

4.20.6 Frame Acquisition Detection.

At each symbol period during the acquisition procedure, the number of PN sequence correlations detected thus far in the current preamble frame is compared to a required number. If the current number of correlations is less than this threshold, the count-down symbol counter R_i is examined to determine if the number of remaining preamble symbols is sufficient to possibly obtain the required number of correlations. If sufficient time remains, the acquisition module decrements the countdown counter and waits for the next set of inputs from the demodulator, thereby continuing the acquisition procedure. If the remaining time is insufficient to obtain the required number of correlations, the counter containing the number of correlations is cleared, thereby restarting the acquisition procedure.

When the number of correlations detected in the current preamble frame equals or exceeds the number required for acquisition, the count-down counter R_i is examined to determine if it is time to enter the data demodulation mode. If not, the counter is decremented and the acquisition procedure continues. This enables the acquisition module to detect additional PN sequence correlations in the current preamble, thereby revising the estimate of the best sample index to apply during the subsequent process of data demodulation.

When the number of correlations is sufficient for acquisition and the symbol counter counts down to the end of the preamble, the acquisition module enters the data demodulation mode. This causes the current estimate of the best sample index to be sent to the demodulator module, where it is used to adjust the matched filter integrate-and-dump timing for data symbol demodulation.

4.20.7 Data Demodulation Mode.

Once the acquisition module enters data demodulation mode, all of the preceding steps involved in preamble acquisition are bypassed while the known number of data symbols are received from the demodulator. The full set of M-ary matched filter outputs is stored in a buffer at each symbol time for subsequent read-out to the following processing modules. There are two buffers for this purpose, which allows for the possibility of contiguous transmission and reception of preamble and data frames. These two buffers operate in a ping-pong manner, with one being read out while the other is being filled, providing demultiplexing of the preamble and postamble symbols and reclocking of data symbols for use by the subsequent processing modules in the link layout.

Once the current write buffer is filled with the data symbol matched filter outputs from the demodulator module, the acquisition module switches back to acquisition mode. The foregoing acquisition procedure is then restarted to search for the preamble at the beginning of the next message, which may begin at any time after the current frame ends.

When the current read buffer is empty, the read and write buffers are swapped and the demultiplexer portion of the acquisition module then attempts to read out demodulated data from the new read buffer. If the next frame was not acquired for any reason, this buffer will be empty with a flag set to denote that condition. Attempting to read out of an empty read buffer causes the missed acquisition flag to be set. The number of missed acquisition attempts is counted and this measurement is included in the print file produced at the end of the simulation run. The acquisition failure rate may also be displayed on the screen during a run using the pick error rate function in the display menu.

4.20.8 Error Rate Measurements.

In COMLNK, error rates are measured continuously during a simulation. While a preamble is being acquired by this module, data from the preceding frame is read out of a buffer and transferred to the subsequent modules. Double buffering allows incoming multiplexed data to be routed into a write buffer while demultiplexed data is output from a read buffer. This introduces a delay of one frame in the acquisition module. Consequently, the number of acquisition attempts is at least one greater than the number of data frames processed through the error counters.

When a preamble is not acquired, null data (all zeros) is read out of the empty buffer and processed through the remaining modules in the link layout. Thus, error-rate measurements are not conditioned on the successful acquisition of a preamble data frame. Instead, the measurements provide the total error rates resulting from a combination of missed acquisitions and normal demodulation and decoding errors.

4.20.9 Signal Timing.

There are at least two ways that signal timing uncertainty can be introduced in a simulation. One is to alter the signal time-of-arrival; another is to alter the receiver timing. In COMLNK, the second method is used. When the user specifies a random timing uncertainty, the receiver's estimate of signal delay is pseudorandomly set within the specified uncertainty range at the beginning of each frame. The difference between the receiver's time estimate and the actual signal time is the delay error, which can be plotted as a function of time by turning on the demodulator plot output file.

The resulting delay error is the timing uncertainty that the receiver is attempting to acquire during each preamble sequence. The acquisition algorithm described above continuously correlates the received signal with the stored preamble sequence at a set of timing positions within the symbol period. When the algorithm decides that a frame has been acquired, a symbol counter is set to indicate the start of the data, and the best timing position within the symbol is chosen for data demodulation. The counter removes the integer portion of the timing uncertainty, while the best timing position removes the fractional portion within the resolution provided by the specified number of timing positions (set by the number of A/D samples per symbol in the demodulator). The delay error is not limited by this resolution, and hence the error will not generally be reduced to zero even with an optimum timing estimate.

Because all of the operations described above take place in the acquisition demultiplexer module, there is no effect on the demodulator's delay estimate. Consequently, a plot of the delay error will show only the initial timing uncertainty at the start of each preamble frame, unless the delay-lock loop is turned on in the demodulator module.

4.20.10 Summary Of Basic Algorithm.

The foregoing acquisition procedure provides symbol synchronization while performing frame acquisition. It offers the capability of operating with any arbitrary timing between message transmissions. However, it suffers from the limitations inherent in the use of hard-decision binary correlators.

Hard-decision correlation produces a loss in symbol timing resolution due to correlator saturation and jitter, which can cause the demodulator symbol timing to be set in a manner that is less than optimum during data demodulation. The loss of symbol timing resolution is potentially significant in strong signal conditions, especially in a fading channel where signal levels may fluctuate dramatically during and after preamble acquisition. Furthermore, in the basic operation of the algorithm described above, the timing estimate is based only on the last correlation, no matter how many timing estimates were actually obtained from the multiple PN sequences.

The design options discussed in the following subsection are intended to alleviate the problems of poor timing resolution and correlation magnitude fluctuations that occur in strong signal conditions and in fading channels. Averaging of the timing estimates, use of a timing threshold, changing to soft-decision correlation, and use of an improved symbol synchronization technique are all available through the user interface to provide better symbol timing estimation with corresponding improvement in data demodulation performance.

4.20.11 Design Options.

A large number of design parameters that affect the operation and performance of this acquisition module are specifiable by the user through the interactive user interface. The number and length of the PN sequences in the preamble are design parameters, along with the bit pattern of each PN sequence. The length of the associated ID words can also be specified, along with their bit patterns. The correlation magnitude threshold, Hamming threshold, and number of correlations required for acquisition are all user-specifiable design parameters. The number of timing positions per symbol is also selectable by way of the number of A/D samples per symbol in the modem data menu.

In addition to specification of these parameters, several other design options can be selected via the user interface. The basic algorithm can be modified to include averaging of the timing estimates obtained from multiple PN sequences. A timing threshold can be specified for the computation of average timing within and across symbol periods in the event of correlation ties. The hard binary correlators can be replaced with soft binary correlators. More fundamentally, the functions of symbol timing synchronization and data frame synchronization can be interchanged. Each of these options is discussed below.

4.20.11.1 Timing Averaging. The basic algorithm discussed thus far uses the last timing estimate from the preamble correlation process to set the integrate-and-dump timing for subsequent data demodulation. A single timing estimate can exhibit considerable error, particularly in a fading channel. It may be quite noisy if it occurs during a signal fade, or it can suffer from poor resolution if it is made during a signal enhancement. When the preamble consists of repeated PN sequences, many timing estimates may be obtained during acquisition, and they can be averaged to reduce the error in demodulation timing.

Averaging of the timing estimates would be straightforward except for the fact that correlations can occur on successive symbol periods. This gives rise to the need for a flag to indicate which symbol period corresponds to the beginning of data at the end of the preamble. The averaging procedure must properly set this flag at the end of the preamble, based on the individual timing estimates accumulated during the preamble. This is handled using a bipolar timing estimate, where a positive value indicates that the best timing occurs in the current symbol period, and a negative value points to the previous symbol period.

The accumulation of these bipolar estimates is referenced to the symbol counter R_i, which is set by the index of the ID word following each PN sequence by means of the counter R_n (Eq. 4.10). Because correlations can occur singly or in pairs on successive symbol periods, it is sometimes necessary to reset the counter during the course of preamble acquisition. Whenever such a reset is performed, the accumulated timing estimates must be re-referenced to the new counter value. Because it is only necessary to reset the counter R_i when it is initially set one count too large, the timing accumulator is re-referenced simply by adding an appropriate positive value equal to the number of timing positions times the number of preceding correlations in this preamble.

At the end of the preamble, when the data demodulation mode is about to be entered, the timing accumulation is divided by the number of valid correlations to provide the best timing estimate. If the average timing is positive, the current/previous symbol flag is cleared, indicating that the best timing falls within the current symbol period. If the average timing is negative, the flag is set indicating that the best timing falls within the previous symbol period. The negative value is then converted to a positive sampling index by adding the number of sample times per symbol, K.

4.20.11.2 Timing Threshold. One of the problems encountered with the preamble acquisition procedure, which exists whether or not timing averaging is used, is that hard binary correlators tend to saturate under strong signal conditions. Even when saturation does not occur, hard binary correlators often exhibit considerable magnitude fluctuation or jitter near the correlation peak, especially in a fading channel.

Both correlator saturation and jitter can significantly degrade the timing resolution. This degradation can be alleviated to some extent through the introduction of a threshold in forming timing estimates. The timing threshold can be set at or above the correlation detection threshold. When correlations exceed the detection threshold but are less than the timing threshold, the peak correlation values are used as before in forming the timing estimate.

When correlations exceed the timing threshold, they are considered equal when forming the timing estimate. With reference to Equations 4.9 and 4.11, all correlations at or above the timing threshold are considered equal to the maximum correlation magnitude when finding the minimum and maximum sample indexes and computing the resulting average index.

4.20.11.3 Soft Correlation. Another method of alleviating timing degradation due to correlator saturation and jitter involves replacing the hard binary correlators with soft-decision correlators. Soft binary correlation is performed using the two competing M-ary FSK symbol metrics that provide the hard binary decisions during preamble search. These two symbol metrics are the matched filter outputs for the two specific frequency deviations used to transmit binary zeros and ones during the preamble (see Equations 4.7 and 4.8).

Let F_{0k} and F_{1k} represent these two metrics, respectively, at the k-th sample time. Then soft-decision correlation accumulations can be formed at each sample time by accumulating the difference between these two metrics, with the sign of the difference depending on whether a zero or a one is present at that bit position in the known preamble sequence. Therefore, the quantity F_{0k}-F_{1k} is added to the soft correlation accumulation when a binary zero is expected, and F_{1k}-F_{0k} is added when a one is expected.

When soft correlation is selected, the only other design parameter that must be changed is the threshold used to detect a possible correlation. For soft correlation, the detection threshold is given in terms of the signal-to-noise ratio in the matched filter outputs. A nominal value of five is suggested for an initial detection threshold. The user should experiment to determine the optimum threshold value.

4.20.11.4 Accumulator Sync. Another design variation reverses the order in which symbol synchronization and frame synchronization are performed. In the design options described thus far, frame synchronization is effectively performed first since preamble sequence correlation is performed at each demodulator sampling time. The sample time that produces the maximum correlation magnitude is then declared the best timing position, thereby establishing symbol synchronization for subsequent data demodulation.

However, symbol synchronization does not require knowledge of the transmitted data stream, only the timing of symbol transitions, and therefore can be accomplished prior to preamble correlation. Symbol timing can be estimated by accumulating the positive difference between the two competing M-ary FSK filter metrics at each sample time. Thus, the difference between the values of F_{0k} and F_{1k} at the k-th sample time are accumulated as follows:

$$\begin{aligned} S_k &= S_{k-1} + F_{0k} - F_{1k} \qquad , \qquad F_{0k} > F_{1k} \\ &= S_{k-1} + F_{1k} - F_{0k} \qquad , \qquad F_{1k} > F_{0k} \end{aligned}$$

The similarity between this symbol timing accumulation and the soft binary correlation process just described is immediately apparent. In effect, this timing accumulation is a soft-decision correlation of the matched filter metrics with the detected demodulated binary sequence. It is a measure of the signal energy in the matched filter outputs. The sample index that produces the largest timing accumulation is likely to be the one most closely aligned with the received symbol timing. To prevent numerical overflow, when the maximum accumulation exceeds a rather arbitrarily selected threshold (currently set at 1024), all accumulators are reduced by a factor of two via a binary right shift.

While these timing accumulations are being formed, the corresponding hard binary decisions are shifted into a set of binary correlation registers, just as described earlier for the basic algorithm. Once per symbol period, when a full set of timing accumulations have been updated, the binary sequence corresponding to the sample index yielding the largest timing accumulation is correlated with the stored preamble sequence. If the result of this hard binary correlation exceeds the specified correlation magnitude threshold, it is used in the preamble acquisition process. The remainder of the acquisition algorithm is essentially unchanged from that described above.

The only design parameter that should be reexamined when this revised symbol synchronization technique is selected concerns the specific bit patterns chosen for the ID words appended to the PN sequence repetitions. Patterns that contain long runs of binary zeros or ones should be avoided. Such patterns provide no symbol timing information and can degrade the performance of the symbol synchronization algorithm.

4.20.12 Diagnostic Output.

Because of the complexity of this acquisition algorithm and its many options, it may be desirable to occasionally view its operation in detail. This can be accomplished during a simulation run by first opening the display menu and then pressing the F11 function key to open a diagnostic display window. A single press of F11 produces a continuous display without interrupting the run. A second press of F11 causes the program to pause at the end of each preamble frame before proceeding to the next frame acquisition. A third press of F11 causes the program to pause after each valid correlation. Pressing any other key allows program operation to continue until the next pause. The action of the F11 keypresses is reversed by pressing F10, which reduces the amount of pause one level at a time, until the diagnostic display window is closed.

When open, the contents of the diagnostic display window are somewhat cryptic to enable a maximum amount of information to be conveyed in a limited space. The first column in the screen window is a count of the number of valid correlations detected thus far in the preamble frame. The second column is the counter R_i (labeled LI in the display). If this counter is reset because of the correlation, the value on display is after that reset but prior to the decrement by one that always occurs. The third column, labeled HW, is the ID word Hamming distance.

The next eight columns are the correlation magnitudes (or timing accumulations in the case of accumulator sync) for up to eight sample times during the symbol period. The column second to the right, labeled TP, shows the current estimate of the best sample time separated by a slash from the current setting of the current/previous symbol flag. The right-most column is a hexadecimal display of a logic flow bit pattern that defines which branch of the code was executed for that correlation.

Additional information is conveyed by the row number in the screen display. The row number is computed from the ID word index used to set the counter R_n, where the first index gives row 1, the second index gives row 3, and so on until the eighth index gives row 15. If there are more than eight PN sequences in the preamble, subsequent indexes cause the window to scroll upward. The row number is then incremented by one if the flag denoting a correlation on the preceding symbol is set. This enables pairs of correlations to be displayed and indicates which ID word passed the Hamming test on each correlation.

If a correlation occurs out of sequence, *i. e.*, an incorrect ID word passes the Hamming test, it may either overwrite a previous correlation, or a subsequent correlation may overwrite it. In either case, if a correlation is overwritten, the one on display will be highlighted in red.

This acquisition algorithm, with its various options, is implemented in the ACQBTX and ACQBRX modules. The state variable datasets of these modules are set up by initialization routines IACQBT and IACQBR.

4.21 FREQUENCY-SHIFT KEYING MODULATION.

The FSK modulator in COMLNK is capable of simulating conventional unhopped FSK, frequency-hopped FSK, and independent-tone FSK modulation. This module accepts the stream of M-ary symbols to be transmitted and sets the frequency control words for the transmitter frequency synthesizer.

The initialization routine for the modulator performs a number of functions that are involved in the design of 32-bit numerically-controlled oscillators (NCOs). NCOs are used in both the modulator and demodulator to generate the carrier and clock frequencies (see Figure 4-1).

The resolution of the clock NCO is set such that one quantum in the NCO control word is equal to the channel symbol rate divided by 2^{24}. The clock resolution may be set somewhat coarser if necessary to accommodate the A/D sampling rate.

The resolution of the carrier NCO is also set such that one quantum in the NCO control word is equal to the channel symbol rate divided by 2^{24}. The carrier resolution may be set somewhat coarser if necessary to accommodate the hopping bandwidth (if any), the maximum FSK modulation tone deviation, or the A/D sampling rate. In no case is the NCO resolution worse than the channel symbol rate divided by 2^{10}.

Each M-ary symbol to be transmitted is converted to a frequency offset from the nominal carrier frequency. As noted above, the carrier can either be hopped or unhopped. If frequency hopping is enabled, there can be one or more modulation symbols per hop period. At the beginning of each new hop period, the frequency control word for the frequency-hop synthesizer is generated pseudorandomly within the specified hopping bandwidth.

To enable receiver timing errors to be treated in detail, the FSK modulator module in COMLNK stores the transmitted modulation tone offset and frequency-hop synthesizer control word in two 64-element circular buffers. This provides channel memory to permit the receiver timing to vary unambiguously over a range of 64 symbol periods. This range is sufficient for accurate simulation of receiver time acquisition and time tracking.

The FSK modulator is implemented in run-time routine MFSKTX. The state variable dataset of this run-time module is initialized by routine IMODTX, which also sets the design parameters for the carrier and clock NCOs.

4.22 FSK TONE FILTER DEMODULATOR.

Three types of noncoherent FSK demodulators are available in COMLNK. These are a conventional tone filter bank demodulator, a discrete Fourier transform (DFT) demodulator, and an independent-tone demodulator. Although each type performs noncoherent M-ary FSK demodulation since the carrier phase is unknown, it should be noted that matched filter detection is inherently a coherent process in that it relies on phase coherence within the modulation symbol period.

The first type of FSK demodulator uses a bank of matched filters to detect which modulation tone was transmitted. The tone filter bank demodulator module can be viewed either as a digital simulation of an analog receiver, or as a digital implementation of a receiver that forms a separate baseband channel for each modulation tone.

After the signal is dehopped, if necessary, and down-converted to a convenient intermediate frequency, this type of receiver splits the input waveform into M distinct signal paths, one for each of the M possible frequency shifts. An integrate-and-dump filter is placed in each path. These filters coherently sum the signal plus noise voltage at each signaling frequency over a sampling period.

In a digital implementation, each of the M signal paths is converted to inphase and quadrature baseband channels. These channels are sampled with integrate-and-dump circuits and converted to I and Q digital samples with analog-to-digital converters. The A/D sampling rate is equal to the modulation symbol rate multiplied by an even integer.

Either an analog or digital implementation, involving a bank of M matched filters or M baseband channels, tends to be hardware intensive. However, this design readily accommodates large frequency spacing between modulation tones that would otherwise require very high sampling rates in a DFT implementation. Here, it is only necessary to sample at least twice the modulation symbol rate in each of the M receiver channels.

Higher sampling rates can be specified, if desired. One reason to use an increased sampling rate with the tone filter demodulator arises in an acquisition mode where symbol timing is initially unknown. This enables a preamble acquisition procedure to determine the best timing position during frame acquisition for use in subsequent data demodulation. To provide matched filter detection with faster sampling and arbitrary symbol timing, the tone filter demodulator incorporates boxcar filters in each of the M baseband channels. These filters can be commanded to provide integrate-and-dump sampling over the symbol period at any desired sample time.

Numerous design options are available in the tone filter demodulator. These include the choice of hopped or unhopped waveforms, envelope or square-law detection, filter clipping level or normalization, noise-based or signal-based automatic gain control, type of frequency-lock loop for carrier frequency tracking, and type of delay-lock loop for time tracking.

This demodulator is applicable to unhopped and frequency-hopped FSK waveforms. If frequency hopping is enabled, there can be one or more FSK symbols per hop. Any hopping bandwidth can be specified, limited by the resolution available in a 32-bit NCO. Refer to the discussion of the FSK modulator (Section 4.21) for more information on the NCO.

A design-point value of carrier power-to-noise density ratio, C/N_o, specifies the signal-to-noise ratio for which the quantizer and tracking loops in the demodulator are designed. When the actual incoming signal level is different from the design level, the loop gains or bandwidths depart from their design values. Quantizer bottoming or saturation may also occur. If an AGC with a signal-based detector is operative, it will attempt to hold the signal level near the design point.

If the user does not specify a design-point value of C/N_o, the program uses a default value that corresponds to an A/D sample energy-to-noise density ratio of 3 dB. In any case, the design-point C/N_o is constrained so that the design level of A/D sample energy-to-noise density ratio is not less than -3 dB and not more than +27 dB. Refer to the discussions of the A/D converter (Section 4.1) and AGC (Section 4.2) for more information on signal and noise levels.

Either square-law or envelope detection of the matched filter outputs can be specified. A clipping level can also be specified, which limits the maximum numerical values of the filter outputs. If a matched filter produces an output above this level, it is limited or clipped. Any 8-bit value in the range from one to 255 may be specified. Alternatively, the matched filter outputs can be normalized by their sum at each dump time.

A power scale factor is used to keep numbers that represent power within a similar range as numbers that represent voltage. Because the FSK demodulator is a pure digital (integer) implementation, the A/D samples are integers that represent a signal plus noise voltage. Values representing power result from multiplication of samples. Without a means of scaling, the power values would overflow or saturate. The scale factor can be specified in the range from one to 255, or a program default value can be used.

The tone filter FSK demodulator is implemented in run-time module MFSK1. The AGC, DLL, and FLL tracking loops are implemented in subroutines that are common to all demodulators. These tracking loops are discussed in Sections 4.2, 4.3, and 4.4, respectively. The state variable dataset for the run-time module is initialized by routine IMFSK1.

4.23 FSK DFT DEMODULATOR.

The second type of M-ary FSK demodulator available in COMLNK employs discrete Fourier transform techniques to implement matched-filter detection of the FSK waveform. In this implementation, the incoming waveform is dehopped and down-converted to baseband where inphase and quadrature channels are formed. The I and Q channels are sampled in A/D converters at a relatively high rate.

DFT demodulation requires that there be a sufficient number of A/D samples per channel symbol to resolve the modulation spectrum. Therefore, the A/D sampling rate depends on the number of signaling tones, M, and on the spacing between adjacent tones, Δf. The number of A/D samples per symbol is a user input, as are the number of tones and tone spacing. If the user-specified sampling rate is too low, the program sets the sampling rate to a minimum value that accommodates the signal bandwidth. The minimum value is set so that the sampling loss is about 0.5 dB. Higher sampling rates will reduce the sampling loss at the expense of more computation.

As is true for all of the communications modules in COMLNK, the DFT demodulator is a pure digital (integer) implementation. The integer implementation of the discrete Fourier transform used to form the matched filter outputs in the DFT demodulator is in subroutine IDFT. This is not the classic fast Fourier transform, although it is a reasonably fast and compact algorithm. A feature of this implementation is that the number of A/D samples per symbol needs only to be a multiple of two, not necessarily a power of two.

This demodulator is applicable to unhopped and frequency-hopped FSK waveforms. If frequency hopping is enabled, there can be one or more FSK symbols per hop. Any hopping bandwidth can be specified, limited by the resolution available in a 32-bit NCO. Refer to the discussion of the FSK modulator in Section 4.21 for more information on the NCO.

Numerous design options are available in the DFT demodulator. These include the choice of hopped or unhopped waveforms, envelope or square-law detection, filter clipping level or normalization, noise-based or signal-based AGC, sync-symbol or suppressed-carrier FLL for frequency tracking, and sync-symbol or suppressed-carrier DLL for time tracking.

A design-point value of carrier power-to-noise density ratio, C/N_o, specifies the signal-to-noise ratio for which the quantizer and tracking loops in the demodulator are designed. When the actual incoming signal level is different from the design level, the loop gains or bandwidths depart from their design values. Quantizer bottoming or saturation may also occur. If an AGC with a signal-based detector is operative, it will attempt to hold the signal level near the design point.

If the user does not specify a design-point value of C/N_o, the program uses a default value that corresponds to an A/D sample energy-to-noise density ratio of 3 dB. In any case, the design-point C/N_o is constrained so that the design level of A/D sample energy-to-noise density ratio is not less than -3 dB and not more than +27 dB. Refer to the discussions of the A/D converter (Section 4.1) and AGC (Section 4.2) for more information on signal and noise levels.

Either square-law or envelope detection of the matched filter outputs can be specified. A clipping level can also be specified, which limits the maximum numerical values of the filter outputs. If a matched filter produces an output above this level, it is limited or clipped. Any 8-bit value in the range from one to 255 may be specified. Alternatively, the matched filter outputs can be normalized by their sum at each dump time.

A power scale factor is used to keep numbers that represent power within a similar range as numbers that represent voltage. Because the demodulator is an integer implementation, the A/D samples are integers that represent a signal plus noise voltage. Values representing power result from multiplication of samples. Without a means of scaling, the power values would overflow or saturate. The scale factor can be specified in the range from one to 255, or a program default value can be used.

Carrier frequency and time tracking can be done on sync symbols or data symbols or both, at the user's option. Time tracking is done using the early-late DFT filter DLL measurement algorithm (Bogusch, 1990). Frequency tracking is done using either the half-tone offset filter FLL measurement, or the early-late DFT filter FLL measurement algorithms.

An additional feature is available in this DFT demodulator, namely an algorithm for automatic frequency ambiguity detection and correction. This enables the demodulator to utilize suppressed-carrier time and frequency tracking even when the initial frequency offset is large compared to the FSK tone spacing. Without an ambiguity resolution scheme, a suppressed-carrier tracking loop will pull into lock at one of the M ambiguous tracking states, M-1 of which are incorrect.

The ambiguity resolution algorithm employed here is that described by (Bogusch, 1990). The algorithm operates by smoothing the difference between the outputs of filters centered above and below the M-ary modulation band in a single-pole digital filter to detect false lock. The time constant of this filter controls how rapidly the algorithm responds when the FLL locks at a multiple of the tone spacing. This time constant is a user-specifiable design parameter. It should be chosen to provide reasonably fast response while minimizing the probability of false alarm.

The frequency ambiguity detection threshold is used to test the output of the filter to determine if the FLL has locked at a multiple of the M-ary FSK tone spacing. As with other detection thresholds, this value is entered on a scale of zero to one and converted internally to an appropriate integer. A value of 0.5 is suggested. When false lock is detected, the carrier NCO is given a discrete shift equal to the FSK tone spacing. Successive shifts may be necessary if the initial lock point is in error by two or more tone spacings. The layout data file FHFSK.DAT on the distribution diskettes provides an example of the operation of this frequency ambiguity detection and correction technique.

The DFT demodulator is implemented in run-time module MFSK2. The AGC, DLL, and FLL tracking functions are implemented in subroutines that are common to all of the demodulators. These tracking loops are discussed in Sections 4.2, 4.3, and 4.4, respectively. The state variable dataset for the run-time module is initialized by routine IMFSK2.

4.24 INDEPENDENT-TONE FSK DEMODULATOR.

The third type of M-ary FSK demodulator available in COMLNK enables the use of independent-tone modulation. This is a form of frequency-hopped FSK that does not exhibit any discernible relationship between the signaling frequencies that correspond to the M possible modulation symbols.

With conventional M-ary FSK, the signaling frequencies are at known offsets relative to the center carrier frequency. With independent-tone FSK there is no such relationship, and indeed the concept of a carrier frequency becomes tenuous. Instead, the modulator can be thought of as having M separate, independent pseudorandom frequency-hop synthesizers. As each M-ary data symbol is transmitted, the value of M selects one of the synthesizers and routes its frequency to the transmitter. Thus, an observation of the transmitted frequency provides no information about the other frequencies that would have been transmitted with different data values. The demodulator contains the same set of M independent pseudorandom frequency synthesizers that must, of course, be synchronized in time and frequency. The synchronization requirements are no different than for any other FH/FSK system. The demodulator forms a bank of M matched filters, one for each of the possible transmitted frequencies. The largest output provides the hard decision as to which data symbol was transmitted. The set of M matched filter outputs provides the soft decision metrics.

Except for the A/D converters, implementation of the independent-tone FH/FSK demodulator is identical to the tone filter bank demodulator discussed in Section 4.22. The A/D converters in the independent-tone demodulator use the M frequency synthesizers to form the center frequency of each matched filter. In effect, there are M distinct receiver front ends.

While hardware intensive, this modulation format offers several advantages. First, it renders a frequency follower jammer ineffective. In fact, such a jammer is more likely to improve performance than it is to degrade it. If the jammer is able to follow the hopped signal, it will place energy around the same frequency that contains the signal, without placing energy at any of the other frequencies that would have been transmitted with a different symbol (assuming that the jammer does not know the M pseudorandom hopping sequences). The most effective jamming strategy is likely to be partial-band noise.

Another advantage of the independent-tone waveform is that it provides robust performance in a wide range of signal scintillation conditions, including fast fading and frequency selective fading. Fast fading is mitigated because of the large effective tone spacing produced by independent tone synthesizers. Frequency selective fading is mitigated by the fact that signal energy that arrives at large delays does not produce interference because the receiver has hopped to a new frequency. These advantages in scintillation can be obtained using more conventional FH/FSK as long as a large tone spacing is used with a large hop bandwidth, together with one symbol per hop. These conditions are assured with independent-tone FSK.

Except for the fact that the independent-tone waveform is inherently frequency hopped with only one symbol per hop, other design options and capabilities of this module are the same as discussed for the tone filter bank FSK demodulator. The independent-tone FSK demodulator is implemented in run-time module MFSK1. The state variable dataset for the run-time module is initialized by routine IMFSK1.

4.25 PHASE-SHIFT KEYING MODULATION.

The PSK modulator in COMLNK is capable of simulating several different forms of phase-shift keyed waveforms. These include binary PSK (BPSK), quaternary PSK (QPSK), and offset QPSK (OQPSK), sometimes referred to as staggered QPSK. In addition, the data in each of these waveforms can be absolutely phase encoded for coherent demodulation (CPSK), or encoded in carrier phase transitions for differentially coherent demodulation (DPSK).

The PSK modulator accepts a stream of binary symbols and forms a modulation symbol appropriate for the specified type of waveform. For coherent BPSK, the input binary symbols themselves provide the phase encoding (zero translates to no phase shift, while one translates to π radians phase shift). Other modulation choices require either differential encoding or formation of quaternary symbols or both. Note that a binary-to-M-ary converter is not needed for QPSK. The conversion from binary to quaternary symbols is performed automatically by the modulator module.

As for all of the modulators, the initialization routine for the PSK modulator performs a number of functions that are involved in the design of 32-bit numerically-controlled oscillators. NCOs are used in both the modulator and demodulator to generate the carrier and clock frequencies.

The resolution of the clock NCO is set such that one quantum in the NCO control word is equal to the channel symbol rate divided by 2^{24}. The clock resolution may be set somewhat coarser if necessary to accommodate the A/D sampling rate.

The resolution of the carrier NCO is also set such that one quantum in the NCO control word is equal to the channel symbol rate divided by 2^{24}. The carrier resolution may be set somewhat coarser if necessary to accommodate the A/D sampling rate. In no case is the NCO resolution worse than the channel symbol rate divided by 2^{10}.

To enable receiver timing errors to be treated in detail, the PSK modulator in COMLNK stores the transmitted modulation symbols in a 64-element circular buffer. This provides channel memory to permit receiver timing to vary unambiguously over a range of 64 symbol periods, which is sufficient for accurate simulation of PSK time acquisition and time tracking.

The PSK modulator is implemented in run-time routine MPSKTX. The state variable dataset of the run-time module is initialized by routine IMODTX, which also sets the design parameters for the carrier and clock NCOs.

4.26 BPSK/QPSK/OQPSK DEMODULATOR.

The demodulator module for unhopped PSK waveforms accepts all modulation waveforms that can be produced by the PSK modulator discussed in the preceding section. Thus, this module performs demodulation of unhopped BPSK, QPSK, or OQPSK waveforms, each of which may be absolutely phase encoded or differentially encoded.

The incoming waveform is sampled in I and Q baseband channels at a rate that is set at twice the modulation symbol rate. Thus, with BPSK and QPSK modulation, two A/D sample pairs are formed per symbol. With OQPSK modulation, the sampling rate is equivalent to one A/D sample pair for each channel bit period because of the staggered bit timing in the I and Q channels.

A full range of tracking functions is available. These include automatic gain control, delay-lock time tracking, frequency-lock carrier tracking, and phase-lock carrier tracking. Composite FLL/PLL carrier tracking is also available. Each carrier loop has an associated lock detector. The algorithms used to implement the tracking error measurements, tracking loops, and associated phase and frequency lock detectors are those described by (Bogusch, 1990).

A design-point value of carrier power-to-noise density ratio, C/N_o, specifies the signal-to-noise ratio for which the quantizer and tracking loops in the demodulator are designed. When the actual incoming signal level is different from the design level, the loop gains or bandwidths depart from their design values. If an AGC with a signal-based detector is operative, it will attempt to hold the signal level near the design point.

If the user does not specify a design-point value of C/N_o, the program will use a default value that corresponds to an A/D sample energy-to-noise density ratio of 3 dB. In any case, the design-point C/N_o is constrained so that the design level of A/D sample energy-to-noise density ratio is not less than -3 dB and not more than +27 dB. See the discussions of the A/D converter (Section 4.1) and AGC (Section 4.2) for more information on the design signal and noise levels.

When operating in an OQPSK mode, this demodulator module provides the option of automatic maintenance of channel integrity. Channel inversions occur in a suppressed-carrier QPSK demodulator when the phase tracking loop slips between adjacent phase lock states, separated by $\pi/2$ radians. Phase slips are a common occurrence in a fading channel. If they do not occur too rapidly, the staggered channel timing in an OQPSK waveform can be used to automatically detect and correct the resulting channel inversions.

The technique for maintaining OQPSK channel integrity is described by (Bogusch, 1990). On-time and mid-bit accumulator outputs are differenced and smoothed in a single-pole digital filter to detect channel inversions. The filter time constant controls the speed of response to an inversion. Small values of the time constant provide rapid response but also increase the probability of false detections. The inversion detection algorithm can be disabled by specifying a zero value for the time constant. When enabled, a specified threshold level is used to determine if the channels are inverted and therefore should be swapped. Because the inversion detection measurement is bipolar, a zero threshold level may be specified.

In all modes of operation, a quantizer scale factor is used to convert the quantized output of the PSK demodulator to the number of bits used in subsequent soft-decision decoding of an error-correction code, if any. The scale factor can strongly affect decoder performance and must be optimized experimentally. A default value of twice the design-point one-sigma noise quanta level is supplied by the program. The user should try higher and lower values in the range from one to 255 to determine the optimum setting.

A quantizer exponent provides the option of using nonlinear scaling in forming the output symbol metrics. The allowable range for this parameter is zero to 3.5. A value of zero (or unity) provides linear quantization.

The unhopped PSK demodulator is implemented in run-time module MPSK1. The AGC, DLL, FLL, and PLL tracking loops are implemented in subroutines that are common to all demodulators. The tracking loops are discussed in Sections 4.2, 4.3, 4.4, and 4.5, respectively. The state variable dataset for the run-time module is initialized by routine IMPSK1.

4.27 PN SPREAD-SPECTRUM MODULATION.

The pseudonoise (PN) modulator in COMLNK simulates binary phase-shift keyed (PN/BPSK) direct-sequence spread-spectrum waveforms. A variety of PN code sequences is available, including all of the GPS clear/acquisition (C/A) codes. Short maximal-length sequences and ideal (very long) PN sequences can also be selected. The data in each waveform can be absolutely encoded in the carrier phase for coherent CPSK demodulation, or differentially encoded in carrier phase transitions for differentially coherent DPSK demodulation.

The PN/BPSK modulator accepts a stream of input binary data symbols and forms modulation symbols appropriate for the specified type of waveform. For coherent BPSK, the data symbols themselves provide the phase encoding (zero translates to no phase shift, while one translates to π radians phase shift). For differential BPSK, the data symbols are encoded in phase transitions where zero produces no change in the carrier phase, while one produces π radians phase change.

As for the other modulators, the initialization routine for the PN modulator performs a number of functions that are involved in the design of 32-bit NCOs. Numerically-controlled oscillators are used in both the modulator and demodulator to generate the carrier and clock frequencies, as indicated in Figure 4-1.

The resolution of the clock NCO is set such that one quantum in the NCO control word is equal to the channel symbol rate divided by 2^{24}. The clock resolution may be set somewhat coarser if necessary to accommodate the A/D sampling rate. The resolution actually used is included in the print output.

The resolution of the carrier NCO is also set such that one quantum in the NCO control word is equal to the channel symbol rate divided by 2^{24}. The carrier resolution may be set somewhat coarser if necessary to accommodate the A/D sampling rate. In no case is the NCO resolution worse than the channel symbol rate divided by 2^{10}.

When a nonideal (finite length) PN code is specified, the modulator computes and stores the entire code chip sequence. This enables the demodulator A/D converter to perform a chip-by-chip correlation between transmitted and local reference code sequences for any amount of error in receiver clock timing. Ideal PN code sequences are approximated by the triangular correlation function that results when very long pseudorandom codes are implemented.

As for the other modulators, the PN/PSK modulator stores the transmitted data symbols in a 64-element circular buffer to enable receiver timing errors to be treated in detail. This provides channel memory to permit receiver timing to vary unambiguously over a range of 64 data symbol periods, which is more than sufficient for accurate simulation of time acquisition and tracking.

The PN/PSK modulator is implemented in run-time routine MPSKTX. The state variable dataset of the run-time module is initialized by routine IMODTX, which also sets the design parameters for the carrier and clock NCOs and calculates and stores the PN chip sequence for finite-length codes. The calculation of PN chip sequences is performed using subroutines CACODE and MAXIML.

4.28 PN SPREAD-SPECTRUM DEMODULATOR.

4.28.1 Introduction.

The demodulator module for PN direct-sequence spread-spectrum PSK (PN/PSK) signals accepts the modulation waveforms that can be produced by the PN/PSK modulator discussed in the preceding section. Thus, this module performs demodulation of binary PSK spread-spectrum waveforms wherein the data may be absolutely encoded in the carrier phase, or differentially encoded in carrier phase transitions.

A variety of PN code sequences may be used for spectral spreading, including any of the GPS C/A codes. Other PN codes that can be used include short maximal-length sequences and ideal (very long) PN sequences.

The PN modem can be configured via the user interface to be either a generic PN receiver, or an accurate simulation of a single-channel GPS L1 C/A receiver. In any case, the PN receiver can be specified to start either in a stable tracking mode (as for the other modems in COMLNK), or in a cold-start signal acquisition mode. Therefore, in addition to the gain control and carrier tracking functions available with the other modems, the PN receiver includes PN code tracking and signal acquisition functions.

A detailed description of a specific configuration of this receiver is given by (Bogusch, 1993). The COMLNK module incorporates all of the features described in the reference. In addition, it enables most of the design parameters to be changed by user input. An overview of each of the functional elements is presented here. More detailed information is provided by the source code listings, which can be accessed using the Info menu.

In all operating modes, the first processing step involves correlation of the incoming spread-spectrum waveform with the local PN reference waveform. The received waveform may be corrupted by noise and propagation disturbances, including amplitude and phase scintillation together with multipath delay spread in frequency selective channels. The received signal also may have any Doppler shift and timing offset relative to the local reference.

With finite-length PN sequences, including the GPS C/A codes, correlation of the incoming waveform with the local reference is performed by means of a precise chip-by-chip correlation of the received and local PN codes. This ensures accurate treatment of the sidelobe structure of short PN codes, including the effects of relatively high-rate A/D sampling that may extend over less than a full code sequence. Correlation of very long PN codes, which usually exhibit negligible sidelobe structure, is approximated by means of an ideal triangular correlation function.

In either case, when multipath disturbances are present, correlations are performed on each delay path by which the signal reaches the receiver. The resulting voltages are combined according to the time-varying amplitude and phase response of each path. This provides a highly accurate simulation of the effects of various signal disturbances that can degrade the PN code correlation process. Hence, time delay and Doppler shift arising from platform dynamics and total electron content (TEC) variations are accounted for, together with the effects of signal amplitude and phase scintillation, including the waveform distortion and sidelobe enhancement produced by frequency selective channels.

For a thorough discussion of frequency selective effects on PN code correlation, together with graphical examples provided by a predecessor of this simulation, see (Bogusch, *et al.*, 1981).

After passing through the code correlator, implemented in routine ADCPN and its subroutines, the incoming waveform is sampled and digitized in I and Q baseband channels. The A/D sampling rate can be set at a multiple of the modulation symbol rate, from twice to 256 times the symbol rate. Thus, two to 256 A/D sample pairs are formed per modulation symbol. This enables a variety of demodulation and tracking options to be selected, including sub-bit data demodulation discussed later in this section.

4.28.2 Operating States.

The PN demodulator may cycle through seven different operating states, depending on the initial mode and the signal conditions. These operating states, numbered internally from 0 to 6, are listed in Table 4-4. The operation of the code and carrier correlators is indicated for each state.

Table 4-4. PN receiver operating states.

State	Code Correlator	Carrier Correlator
0	Acquisition, fast noise AGC	Wait for code acquisition
1	Acquisition, code search	Wait for code acquisition
2	Acquisition, sidelobe rejection	Wait for code acquisition
3	Delay-lock loop pull-in and AGC	Wait for code acquisition
4	Delay-lock loop tracking and AGC	Fast signal AGC
5	Delay-lock loop tracking and AGC	Frequency pull-in and bit sync
6	Delay-lock loop tracking and AGC	FLL/PLL tracking, AGC, data demod

The code correlator performs the functions of acquiring and tracking the PN code timing of the received signal. The carrier correlator performs the functions of acquiring and tracking the signal frequency (and phase if coherent processing is selected), as well as data demodulation.

If the receiver is initiated in a cold-start acquisition mode, it begins in state 0 to acquire the PN code. The first step is to establish the noise level using a fast AGC algorithm. Then a two-dwell code search procedure is initiated in state 1. Because a short PN code can be acquired on a sidelobe, a sidelobe rejection function may be invoked in state 2. Then in state 3, a signal AGC function is initiated with delay-lock loop pull-in.

The code correlator attains a stable tracking condition in state 4, at which time the carrier correlator begins to acquire the signal carrier frequency. To set the gain in the carrier correlator, a fast signal AGC is performed in state 4, followed by frequency pull-in and data bit synchronization in state 5. The carrier correlator attains a stable tracking condition in state 6, whereupon data demodulation is initiated. State 6 is the final operating state; the PN receiver remains in this state once it is reached.

If the receiver is initiated in a tracking mode, states 0 through 5 are not executed and state 6 is entered immediately. This assumes that the received signal time and frequency are known to within the pull-in ranges of the code and carrier tracking loops, and that code and data bit synchronization have already been achieved.

4.28.3 PN Code Acquisition And Tracking.

4.28.3.1 Noise AGC. The first step in signal acquisition is the estimation of the noise level in the receiver. The noise level must be known so that the detection threshold can be set properly for code search and verification. The noise estimate is obtained using a fast binary search algorithm. With reference to the discussion of A/D converters, the 8-bit gain control word for the code correlator A/D converter is set initially to the middle of its range. This involves setting the high-order bit and clearing the seven low-order bits of the control word; hence, the initial setting of the gain control word is 128.

The local PN code generator is slewed at a rapid rate to avoid correlating with the signal. A noise power measurement is formed by noncoherently accumulating a specified number of A/D samples, and the result is compared to a threshold level. The threshold is specified by the design-point noise level in the A/D output. If the measurement is below the threshold, the gain control word is increased by setting the next significant bit. Otherwise, the control word is reduced by clearing the current bit and then setting the next significant bit. Therefore, at the end of the first measurement the gain control word is either 128+64 or 64.

The same procedure is repeated with the new setting of the gain control word. After the measurement, the next lower significant bit is set and the current bit is either left set or cleared depending on whether the measurement is below or above the threshold, respectively. At the end of the second measurement there are now four possible values of the gain control word: 128+64+32, 128+32, 64+32, or 32.

This procedure is repeated for seven measurements, at which point all seven high-order bits of the gain control word have either been set or cleared, and the low-order bit is set. The noise level at the code correlator A/D output is thus adjusted to the design level within the accuracy of the measurements and the resolution of 8-bit gain control. This fast AGC algorithm is implemented in routine AGCFST, which is part of module MPSK2.

4.28.3.2 Code Search. Once the noise level is established, code search is initiated to acquire the timing of the PN code. The code NCO is adjusted to a rate slightly different than the nominal PN code rate. The difference causes the local code timing to gradually retard (or advance, depending on the sign of the difference) relative to the received signal code timing. A power measurement is formed by noncoherently accumulating a predetermined number of A/D samples, and the result is compared to a threshold level.

The threshold is set to provide a specified probability of erroneous detection (false alarm rate). The number of A/D samples used to form the measurement is set such that the local code timing slews one-half the PN code chip period during the measurement interval. If the measurement falls below the threshold, the process continues with another measurement. Successive search measurements therefore occur at relative time intervals of one-half chip. Search may either proceed over the entire PN code sequence, or it may be performed in a sawtooth pattern over a specified number of chips.

If the measurement is above the threshold, the incoming signal may have been detected. A second measurement is then performed to verify the detection or to reject a false alarm. During this verification measurement, the code NCO is set at the nominal PN chip rate so that the relative timing remains unchanged. The verification measurement is formed by noncoherently accumulating a number of A/D samples twice that used in the search measurement, and the resulting power is compared to a higher threshold level. The verification threshold is set to provide a specified false alarm probability.

If the verification measurement is below threshold, the detection is considered a false alarm and is ignored. The code NCO is then reset to the previous slew rate and the search procedure is reinitiated. Code search continues until the code is found. or until the elapsed time reaches a specified time-out value. Inability to detect the signal may be caused by an incorrect gain setting. Therefore, if time-out occurs in search, the receiver restarts in state 0 to reset the gain. Code search is then reinitiated.

Once a detection is verified, the PN code is considered to be found and the next receiver state is entered. This two-dwell code search algorithm is implemented in routine ACQCOD, which is part of module MPSK2.

4.28.3.3 Sidelobe Rejection. Short PN codes can produce significant correlation even when the timing is in error by many chips. This is especially true if the A/D sampling interval does not encompass the full code sequence. For example, the GPS C/A codes exhibit rms sidelobe correlation levels around -30 dB when correlated over the full 1023-chip sequence. However, rms levels increase to -26 dB with a 2400 Hz A/D sampling rate (426 chips per sample), with peak sidelobe levels reaching -15 dB. In strong signal conditions, especially in the presence of signal enhancements in a fading channel, this can lead to sidelobe acquisition.

An algorithm to detect sidelobe acquisition and correct the receiver timing is therefore included in the PN receiver. The code NCO is offset from the nominal PN code rate so that the local code timing slews one-half chip per A/D sample. Noncoherent power measurements are formed using two A/D sample pairs. Thus, successive measurements are formed at delay increments of one code chip. Each measurement is compared to a threshold level, set such that the probability of false detection is reduced by a factor of ten from that used in code search.

This process continues until the entire code sequence has been tested, or until a measurement exceeds the threshold. If no measurement exceeds the threshold, the original timing is most likely correct and is left unchanged. If at any time in this process the threshold is exceeded, the corresponding code timing is taken to be correct since the original acquisition was most likely produced by a sidelobe. In either event, the code is now considered to have been properly acquired, and the receiver enters the next state.

Because very long PN codes usually exhibit negligible sidelobes and are treated as ideal, the sidelobe rejection state is bypassed for such codes. This is accomplished by appropriate initialization of the process, which is implemented in routine RJCTSL, a part of module MPSK2.

4.28.3.4 Code Pull-In and Tracking. The final process executed by the code correlator is pull-in to a stable code tracking state. This is accomplished using a delay-lock loop with tau-dithered early-late delay error measurements. A signal-based AGC algorithm is included to maintain the loop parameters near their design values.

The PN code loop alternately switches the code correlator between timings that are one-half chip early and one-half chip late relative to the punctual code. Early and late measurements are formed by accumulating a specified number of A/D samples at each timing, and the pair of measurements are then used to form a delay error measurement.

Three processing options are provided for the delay error measurement. A fully coherent measurement utilizes the coherent accumulation of I-channel samples during the early and late code settings. This option is only available when carrier phase lock is achieved. A quasi-coherent measurement utilizes coherent accumulations of I-channel and Q-channel samples, forming power measurements for the early and late settings. A noncoherent measurement utilizes accumulations of $I^2 + Q^2$ samples over the early and late intervals. In each case, the early and late measurements are differenced to form the delay error measurement. See (Bogusch, 1990) for detailed description of these three techniques.

Delay error measurements are fed into a delay-lock loop that can be first-order, second-order, or third-order. The loop noise bandwidth and other parameters are user inputs. These design values are specified at the loop design point, given in terms of a design-point value of carrier power-to-noise density ratio, C/N_o.

If the user does not provide a design value of C/N_o, a default corresponding to an A/D sample energy-to-noise density ratio of 3 dB is assumed. In any case, the design-point C/N_o is constrained so that the design level of A/D sample energy-to-noise density ratio is not less than -3 dB and not more than +27 dB. See the earlier discussions of the A/D converter (Section 4.1) and AGC (Section 4.2) for more information.

When the actual incoming signal level is different from the design level, the loop parameters depart from their specified values. A signal-based AGC in the code loop attempts to hold the signal level near the design point. The gain measurement is obtained by summing noncoherent power measurements from the early and late code settings. The AGC control value is derived from the design-point value of C/N_o. Together with the design-point noise level in the A/D output, this specifies the design level of A/D output when signal energy is present.

The algorithms used for PN code pull-in, tracking, and AGC are implemented in routine TRKCOD, which is part of module MPSK2.

4.28.4 Carrier Acquisition And Tracking.

4.28.4.1 Signal AGC. Once the PN code is acquired and the code correlator enters its tracking state, acquisition of the signal carrier frequency can commence. Carrier Doppler shift is typically much less than the bandwidth of the spread-spectrum signal. Hence, frequency acquisition can wait until code acquisition is completed if the A/D sampling rate is sufficiently high to accommodate the frequency uncertainty.

The first step in carrier acquisition is to set the gain in the carrier correlator A/D converter. This is done with the same fast AGC algorithm used to set the code correlator gain at the start of code acquisition. The only difference in the carrier correlator is that the gain can be set initially on the basis of signal plus noise. Consequently, no code slew is used (it is not needed and would disrupt code track), and the AGC threshold is set using the design-point value of signal plus noise.

4.28.4.2 Carrier Pull-In and Bit Synchronization. Once the gain is set, the carrier correlator uses a noncoherent frequency-lock loop to acquire the carrier frequency. Frequency error measurements are formed from the cross product of successive I-channel and Q-channel A/D samples. The error measurements are fed into the FLL, which can be either first-order or second-order. The loop order, bandwidth, and damping factor (if second-order) are specified by the user.

During carrier pull-in, bit synchronization is also acquired. This is necessary for cases where the PN code period is shorter than the data bit period. In such cases, as for example with the GPS C/A codes, acquisition of PN code timing does not unambiguously provide data bit timing.

Because a short PN code may repeat several times during a data bit period, it is necessary to determine which A/D sample corresponds to the bit edge.

Bit synchronization is accomplished by forming an N-bin histogram of sign reversal counts, where N is the number of A/D samples per data bit. Sign reversals are detected using the dot product of successive I and Q sample pairs. The signal component of the dot product is negative whenever the phase changes more than 90 degrees between samples. Hence, a negative value is likely due to a data phase transition. Noise, interference, and frequency error may introduce spurious sign reversals, but their effects can be rendered insignificant by accumulating a sufficiently large number of measurements.

At each sampling time, the counter for the current sample index is incremented if the sign of the dot product is negative. When one of the counts exceeds its closest competitor by a specified threshold value, the edge of the data bit is considered to have been found. The corresponding sample index is then set to be the first sample in the bit period, thereby synchronizing the receiver sample timing with data bit transitions.

If bit synchronization is not achieved within a specified period, it is likely that no signal is actually present even though PN code acquisition was deemed successful. This may result from an erroneous gain setting or perhaps from acquisition of a code sidelobe in a multipath condition, for example. In any event, if time-out occurs during bit synchronization the receiver is restarted in state 0 to reset the gain and reacquire the code.

The bit sync time-out time is taken to be one-eighth of that specified for code search time-out, but not less than four times the number of bit periods required to achieve the specified bit sync count threshold value. The algorithms used for carrier pull-in and bit synchronization are implemented in routine ACQCAR, which is part of module MPSK2.

4.28.4.3 Data Demodulation. Once bit synchronization is acquired, the carrier correlator enters its tracking state and commences data demodulation. A number of tracking, demodulation, and gain control options are available. The basic algorithms used to implement the tracking error measurements, tracking loops, and associated phase- and frequency-lock detectors are those described by (Bogusch, 1990). Additional detail can be obtained from examination of the source code in the Info menu.

Carrier tracking and gain control can be performed at one of three degrees of coherency, referred to as robustness levels. The most robust processing option, that which requires the least channel phase coherence, employs individual I and Q baseband A/D samples. The least robust option, requiring the greatest phase coherence, employs coherent accumulations of I and Q samples over the full data bit period. An intermediate option employs accumulations over part of the bit period. The three robustness levels are called baseband, full-bit, and sub-bit processing, respectively.

In each of the three robustness levels, automatic gain control can be specified to operate in one of two signal-based modes (noncoherent or coherent) or in a noise-based mode. All three AGC modes offer the choice of power or envelope detection. Therefore, the user can choose from six AGC options in each of three robustness levels.

Carrier tracking can be performed with a frequency-lock loop, a phase-lock loop, or a combination of the two. These three options are available in each of the three robustness levels. Frequency and phase error detectors are included. If combined FLL and PLL tracking is selected, a composite loop is employed (Cahn, 1977; Bogusch, 1990). The composite loop utilizes both frequency error and phase error measurements until phase lock is acquired. When the carrier loop is in phase lock, the FLL is automatically disabled. The FLL is automatically enabled if phase lock is lost.

The loop order, noise bandwidth, and other parameters are user inputs for both the FLL and PLL. The FLL can be first-order or second-order, while the PLL can be first-order, second-order, or third-order. Either loop can be turned off if desired to provide only frequency or only phase tracking. The loop parameters will vary from their specified values if the incoming signal level departs from its design-point value. A signal-based AGC, if operative, will attempt to hold the signal level and loop parameters near the design values.

Data demodulation can be performed using either full-bit or sub-bit robustness. Full-bit data demodulation is much more common as it utilizes all of the energy in the data bit period. Sub-bit demodulation is an option unique to this PN receiver. When sub-bit demodulation is selected, only a portion of the energy surrounding bit transitions is used to form the demodulation decision. While this degrades performance in noise, it may provide improved performance in fast fading where channel phase coherence is lost over the bit period.

If the transmitted waveform can be changed, low-rate error-correction encoding or symbol repetition are preferable to sub-bit demodulation for mitigation of the effects of fast fading. However, if the modulation waveform is fixed, as in GPS, use of sub-bit demodulation offers a way of avoiding catastrophic failure of data demodulation in fast fading channels.

When sub-bit demodulation is used, the number of sub-bits per modulation bit is a user-specifiable parameter. Demodulation is then performed using the last sub-bit in the preceding bit period and the first sub-bit in the current bit period to compute a dot-product decision variable. The sign of the dot product detects a phase transition, and the magnitude provides a quality measure.

Sub-bit demodulation is therefore the same as DPSK demodulation using just the energy surrounding the data bit edge. While DPSK demodulation is most commonly used with DPSK modulation, it is not so restricted. DPSK demodulation can be used even if coherent (CPSK) modulation is employed at the transmitter. The resulting combination, referred to as RPSK, invokes a differential encoder in the demodulator to convert the transition decisions to a form corresponding to absolute phase encoding. This leaves a polarity ambiguity in the demodulated data stream, which can be resolved with differential encoding/decoding or with an error-detection code that compensates for 180-degree phase ambiguity. An example of the latter is the (32,26) Hamming code used in GPS.

In all modes of operation, a quantizer scale factor is used to convert the quantized output of the PN/PSK demodulator to the number of bits used in subsequent soft-decision decoding of an error-correction code, if any. The scale factor can strongly affect decoder performance and must be optimized experimentally. A default value of twice the design-point one-sigma noise quanta level is supplied by the program. The user should try higher and lower values in the range from one to 255 to determine the optimum setting.

A quantizer exponent provides the option of using nonlinear scaling in forming the output symbol metrics. The allowable range for this parameter is zero to 3.5. A value of zero (or unity) provides linear quantization.

The PN/PSK demodulator is implemented in routine TRKCAR, which is part of the module MPSK2. AGC, DLL, FLL, and PLL are implemented in subroutines that are common to all demodulators. The tracking loops are discussed in Sections 4.2, 4.3, 4.4, and 4.5, respectively. The state variable datasets for these run-time routines are initialized by routine IMPSK2.

4.29 FREQUENCY-HOPPED PSK MODULATION.

Phase-shift keying is usually associated with conventional unhopped carriers. However, PSK modulation can be used in conjunction with frequency hopping. While this combination may appear at first to be somewhat unnatural, it is found to work quite well and offers some advantages in fading channels. The FH/PSK modulator in COMLNK accommodates either BPSK or QPSK modulation. Either waveform can be absolutely encoded or differentially encoded.

The FH/PSK modulator accepts a frame-formatted set of binary symbols, consisting of specified numbers of data symbols and reference symbols. The reference symbols are necessary to establish a phase reference for demodulation of each hop. One or more data symbols preceded by one or more reference symbols may be specified for the hop frame. Hop frame formatting and insertion of reference symbols is performed by a separate module, similar to the sync multiplexer discussed previously. If no hop multiplexer is explicitly included in the link layout, the program will automatically insert one immediately preceding the FH/PSK modulator.

At the start of each new hop frame, the frequency control word for the hop synthesizer is generated pseudorandomly within the specified hopping bandwidth. As with other modulator modules, the modulation symbol and frequency-hop control word are stored in 64-element circular buffers to provide channel memory for accurate treatment of timing errors in the demodulator.

The channel capacity of an FH/PSK modem is set by the frequency-hop rate and the number of information symbols per hop. If the hop rate is set too low to carry the data for a single user, it will automatically be increased by the program at run time. If it is set to a value larger than needed to carry the user data, the modem will automatically be multiplexed with other channels.

As for the other modulators, the initialization routine for the FH/PSK modulator sets design parameter values for 32-bit NCOs, which are used in both the modulator and demodulator to generate carrier and clock frequencies (see Figure 4-1). The resolution of both the clock and carrier NCO is set such that one quantum in the NCO control word is equal to the channel symbol rate divided by 2^{24}. The carrier resolution may be set somewhat coarser if necessary to accommodate the hopping bandwidth, but in no case is the NCO resolution worse than the channel symbol rate divided by 2^{10}.

The FH/PSK modulator is implemented in run-time routine MPSKFH. The state variable dataset for the run-time module is initialized by routine IMODTX, which also sets the design parameters for the carrier and clock NCOs.

4.30 FH/PSK DEMODULATOR.

The demodulator module for frequency-hopped PSK waveforms accepts all of the waveforms provided by the FH/PSK modulator described in the preceding section. Therefore, this module performs demodulation of FH/BPSK and FH/QPSK waveforms, each of which may be absolutely phase encoded or differentially encoded.

The channel capacity of an FH/PSK link is set by the frequency-hop rate and the number of information symbols per hop. If the hop rate is set too low to carry the data for a single user, it will automatically be increased at run time. If it is set to a value larger than needed to carry the user data, the modem will automatically be multiplexed with other channels.

Hence, the FH/PSK demodulator may be used in time-division multiplexed links involving many users, each of which utilizes some fraction of the total number of frequency hops. The extra hop frames added in the TDM process are discarded after demodulation.

After dehopping and down-conversion to baseband, the demodulator samples the inphase and quadrature channels with A/D converters at a rate equal to twice the modulation symbol rate. The samples are stored until all symbols in a hop frame have been received.

Upon receipt of a hop frame corresponding to the user of interest, the A/D samples are used to form tracking error measurements as well as to demodulate the user data. As described by (Bogusch, 1990), demodulation of FH/BPSK waveforms can be done coherently or differentially coherently, regardless of whether absolute or differential phase modulation is employed. Consequently, there are four possible FH/BPSK configurations, corresponding to all four combinations of two modulation techniques and two demodulation techniques, as summarized in Table 4-5.

Table 4-5. FH/BPSK modulation/demodulation options.

Name	Phase Modulation	No. Reference Symbols/Hop	Reference Location	Demodulation Algorithm
CPSK	Absolute	≥ 1	Anywhere in Frame	Coherent (PLL Enabled)
RPSK	Absolute	≥ 1	Anywhere in Frame	Differential (PLL Disabled)
ΔPSK	Differential	1	Preceding Symbol	Coherent (PLL Enabled)
DPSK	Differential	1	Preceding Symbol	Differential (PLL Disabled)

When coherent demodulation is selected, the techniques are called CPSK and ΔPSK depending on whether absolute or differential phase modulation is employed, respectively. With differentially coherent demodulation, the corresponding techniques are referred to as RPSK and DPSK, respectively. Three of these four techniques can be used in unhopped PSK links. The exception is RPSK, wherein absolute phase modulation is combined with differentially coherent demodulation. This technique is generally limited to FH/PSK because it depends on the presence of reference symbols.[1]

The number of reference symbols in each hop frame is a user-specifiable design parameter. With absolute phase modulation (CPSK or RPSK), the optimum number of reference symbols per hop depends on the number of data symbols. With differential phase modulation (ΔPSK or DPSK), a single phase reference is implicit. However, additional pseudo-reference symbols may be specified to obtain a desired total number of symbols per hop.

The specification of the number of data and reference symbols per hop frame can be given with the hop frame multiplexer, or with the FH/PSK modem. If no hop multiplexer is explicitly included in the link layout, one will be inserted automatically before the FH/PSK modulator. If a hop multiplexer is specified, the associated numbers of data and reference symbols override values given with the modem.

All four modulation/demodulation techniques (CPSK, ΔPSK, RPSK, DPSK) are available with FH/BPSK waveforms. Only differentially coherent demodulation is available with FH/QPSK waveforms, limiting the selection to RPSK and DPSK with FH/QPSK.

In any case, a full range of tracking functions is available, including AGC, DLL time tracking, and FLL carrier tracking. Carrier phase estimation replaces PLL carrier tracking when coherent demodulation is used with FH/BPSK. Several methods of estimating carrier phase within each

[1] A variation of RPSK is available in the PN/PSK receiver (Section 4.28) when it is operating in the sub-bit demodulation mode.

hop period have been considered. The technique that provides the best performance, known as the inverse tangent estimator, is used (Bogusch, 1990).

A design-point value of carrier power-to-noise density ratio, C/N_o, specifies the signal-to-noise ratio for which the quantizer and tracking loops in the demodulator are designed. When the actual incoming signal level is different from the design level, the loop gains or bandwidths depart from their design values. If an AGC with a signal-based detector is operative, it will attempt to hold the signal level near the design point. If no design-point value of C/N_0 is specified, the program will use a default value that corresponds to an A/D sample energy-to-noise density ratio of 3 dB. In any case, the design-point C/N_0 is constrained so that the design level of A/D sample energy-to-noise density is not less than -3 dB and not more than +27 dB. See the discussions of the A/D converter (Section 4.1) and AGC (Section 4.2) for more information on the design-point values of signal and noise.

A quantizer scale factor is used to convert the quantized output of the FH/PSK demodulator to the number of bits used in subsequent soft-decision decoding of an error-correction code, if any. The scale factor can strongly affect decoder performance and must be optimized experimentally. A default value of twice the design-point one-sigma noise quanta level is supplied by the program. The user should try higher and lower values in the range from one to 255 to determine the optimum setting.

A quantizer exponent provides the option of using nonlinear scaling in forming the output symbol metrics. The allowable range for this parameter is zero to 3.5. A value of zero (or unity) provides linear quantization.

The FH/PSK demodulator is implemented in run-time module MPSK3. The AGC, DLL, and FLL tracking loops are implemented in subroutines that are common to all demodulators and are discussed in Sections 4.2, 4.3, and 4.4, respectively. The state variable dataset for the run-time module is initialized by routine IMPSK3.

SECTION 5

THE FADING RADIO CHANNEL

When radio frequency signals propagate along paths through the ionosphere, or in mountainous terrain or urban environments, signal fading or scintillation may be observed at the receiver. Fading and scintillation (the terms tend to be used interchangeably) are characterized by random fluctuations in received signal amplitude, phase, angle-of-arrival, and time delay.

Scintillation may result from multipath, scattering, or diffraction. The distinction between these causative mechanisms is often more semantical than physical. Multipath can be produced by scattering, and the term scattering is frequently used interchangeably with diffraction. Diffraction, in turn, can be viewed as being equivalent to the superposition of waves traveling in slightly different directions, and hence can be thought of as micro-multipath.

Ionospheric scintillation arises from random variations in the index of refraction, resulting from ionization irregularities produced by plasma instabilities. The ionization tends to break up into filaments, or striations, aligned with the geomagnetic field. One can visualize striations as long sheets or rods of relatively high electron density embedded in a background of lower electron density, as sketched in Figure 5-1.

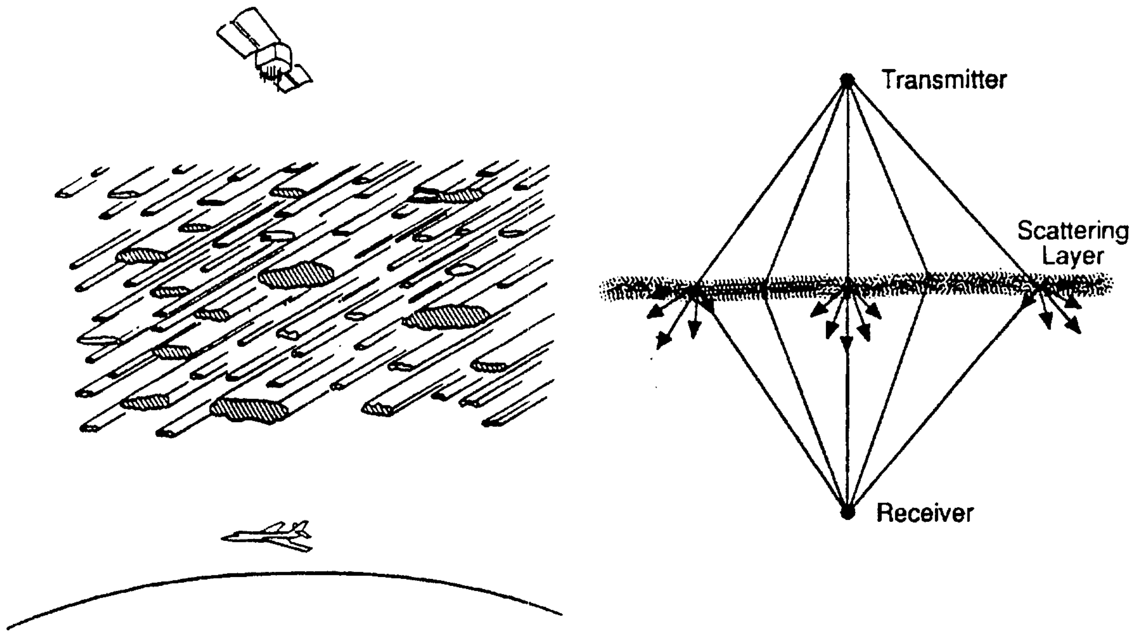


Figure 5-1. Signal propagation through a striated ionosphere.

Consider for a moment an unmodulated wave traversing a region of random fluctuations in the index of refraction. The wave first suffers random phase perturbations due to variations in the phase velocity. As the wave propagates further, diffractive effects introduce fluctuations in amplitude as well as phase, resulting in undesired complex modulation of the carrier.

In the ionosphere, the index of refraction and phase velocity depend on the frequency of the propagating electromagnetic wave. A communications signal encompasses a spectrum of frequencies because of the transmitted data modulation. The frequency-dependent index of refraction causes each spectral component of the signal to experience different phase and amplitude scintillations when propagating along a transionospheric path.

If the resulting differences are minor (*i. e.*, if fading is highly correlated across the signal bandwidth), the propagation channel is said to be nonfrequency selective, or flat fading. When significant statistical decorrelation is observed across the signal bandwidth, the channel is said to be frequency selective. No clear demarcation exists between the two cases; the distinction is merely a matter of degree of selectivity. However, the terms are in common use and provide a convenient framework for discussion and channel modeling.

5.1 FLAT FADING CHANNELS.

When the random fluctuations of each spectral component of the received signal exhibit essentially identical behavior in time, the channel is said to be flat fading. Flat fading occurs when the signal bandwidth is small compared to the channel coherence bandwidth. The effect of the channel can then be represented mathematically by a complex multiplicative factor on the transmitted signal. Thus, if the real transmitted signal is written as

$$s(t) = \mathrm{Re}\{m(t)e^{j\omega t}\}$$

where $m(t)$ is the transmitted data modulation, ω is the carrier angular frequency, and Re{x} denotes the real part of the argument, then the received signal in a flat fading channel can be written as

$$r(t) = \mathrm{Re}\{A(t)e^{j\theta(t)}m(t)e^{j\omega t}\} \tag{5.1}$$

where $A(t)$ is the amplitude modulation imposed by the propagation channel, and $\theta(t)$ is the corresponding channel-imposed phase modulation. The inphase and quadrature-phase (I and Q) components of the channel modulation are $A(t)\cos\theta(t)$ and $A(t)\sin\theta(t)$, respectively. The first-order statistics of the fading channel therefore can be defined either in terms of the amplitude and phase functions, or in terms of the I and Q components.

5.1.1 First-Order Statistics.

Radio wave propagation through regions of index of refraction fluctuations has been studied for many years. An extensive body of data has been accumulated from a wide variety of field experiments, especially in the equatorial and polar regions of the earth's ionosphere. The data have been analyzed to determine the statistical characteristics of scintillation. Distributions of signal amplitude and phase are described by the first-order scintillation statistics. Temporal, spatial, and spectral correlation properties are described by the second-order statistics.

Statistical analyses of experimental data and theoretical results reveal that when scintillation is intense or fully developed, the random signal amplitude is Rayleigh distributed and the random channel phase is uniformly distributed over 2π radians. Hence, in a Rayleigh fading channel the phase function $\theta(t)$ is uniformly distributed over an interval $[0, 2\pi]$, and the amplitude function $A(t)$ is described by the Rayleigh probability density function:

$$f(A) = \frac{A}{\sigma^2}\exp\left[-\frac{A^2}{2\sigma^2}\right] \qquad , \qquad A \ge 0$$

where σ^2 is the variance of each of the I and Q components of the scattered signal voltage. In terms of received signal power $P = A^2$, the value of σ^2 is equal to one-half the mean power, and the Rayleigh distribution becomes

$$f(P) = \frac{1}{P_o} \exp\left[-\frac{P}{P_o} \right] \qquad , \qquad P \geq 0$$

where $P_o = 2\sigma^2$ is the mean received signal power. The cumulative distribution of signal power in a Rayleigh fading channel is then

$$F(P) = \int_0^P f(P')dP'$$
$$= 1 - \exp\left(-\frac{P}{P_o} \right)$$

It is readily seen from this cumulative distribution that in a Rayleigh fading channel, the signal exhibits fades below -10 dB about ten percent of the time, below -20 dB one percent of the time, and below -30 dB one-tenth of one percent of the time.

This strong scattering limit can also be obtained by applying the central limit theorem to the superposition of many randomly scattered waves. The I and Q components of the resulting electric field are found to be zero-mean Gaussian random variables, statistically independent, with equal variances of one-half the mean signal power. These conditions are both necessary and sufficient for Rayleigh statistics, which describe the envelope of a narrowband Gaussian noise process. Thus, the first-order signal statistics for Rayleigh fading are the same as those for narrowband Gaussian noise.

One measure of the intensity of signal scintillation is the S_4 scintillation index. The S_4 index is simply the normalized standard deviation of signal power. In terms of signal voltage amplitude A, the S_4 index is defined as

$$S_4 = \left[\frac{\langle A^4 \rangle - \langle A^2 \rangle^2}{\langle A^2 \rangle^2} \right]^{1/2}$$

where angle brackets denote statistical averages.

In intense scintillation conditions, the value of S_4 saturates at unity, which is a necessary (but not sufficient) condition for Rayleigh statistics. The value of S_4 is zero in nonfading channels. Values of S_4 between zero and unity correspond to weak to moderate scattering, where signal fading is less severe than Rayleigh. Values of S_4 somewhat greater than unity are occasionally measured; such values are usually associated with focusing effects that often accompany multipath conditions. For most systems, the worst condition for a single one-way propagation path is Rayleigh fading.

5.1.2 Second-Order Statistics.

Relative motion between the propagation path and scattering region, due to path movement or scattering region movement or both, causes the channel to vary with time. The resulting time-varying amplitude and phase modulation may have rates ranging from slow to very fast. The fading rate depends on the relative path velocity, intensity of the scattering irregularities, and wave frequency. A precise measure of fading rate is provided by the scintillation decorrelation time, which is a parameter involved in the second-order signal statistics.

The signal decorrelation time is denoted by τ_o and is defined at the $1/e$ point on the time autocorrelation function of the complex channel modulation. Writing the complex channel modulation as $E(t) = A(t) \exp[j\theta(t)]$, the autocorrelation function is defined by

$$R(\tau) = \langle E^*(t)\, E(t+\tau) \rangle$$

where the asterisk denotes complex conjugation. The decorrelation time is now defined by the relationship

$$R(\tau_o) = e^{-1} R(0)$$

Because small values of τ_o correspond to fast fading while large values correspond to slow fading, the decorrelation time is an inverse measure of the fading rate.

Another measure of fading rate is the Doppler spread. In this context, Doppler spread refers to spectral spreading of the signal resulting from time variation of the channel-imposed amplitude and phase modulation. The more rapid the scintillation, the greater the Doppler spread. Thus, small values of τ_o correspond to large values of Doppler spread, and conversely. Doppler spread is defined by the bandwidth of the scintillation power spectrum *S(f)*, which is given by the Fourier transform of the time autocorrelation function:

$$S(f) = \int_{-\infty}^{\infty} R(\tau) e^{-j2\pi f\tau}\, d\tau$$

where the Doppler frequency f is measured relative to the signal carrier frequency.

Measurements of the fading power spectrum have been made with experimental data from satellite transmissions and with signal realizations from multiple phase screen calculations. The spectrum also has been investigated analytically. In slow fading conditions, the spectrum is generally found to exhibit a power-law dependence on Doppler frequency, with an f^{-4} dependence being representative of the observed spectra. In fast fading conditions, the fall-off with Doppler frequency is more rapid, limiting to a Gaussian spectrum when the fading is quite fast.

An f^{-4} form for the fading spectrum is particularly convenient for channel simulation because it can be synthesized using two-pole filters, which are easily implemented in hardware or software. When the two-pole filters are formed using pairs of cascaded single-pole RC filters, the normalized f^{-4} spectrum is given by

$$S(f) = \frac{4\tau_{RC}}{1 + 2(2\pi\tau_{RC} f)^2 + (2\pi\tau_{RC} f)^4}$$

where τ_{RC} is the time constant of each of the single-pole RC filters. The relationship between this time constant and the signal decorrelation time τ_o is obtained from the autocorrelation function, given by the Fourier transform of this power spectrum:

$$R(\tau) = \left(1 + \frac{|\tau|}{\tau_{RC}}\right) \exp\left(-\frac{|\tau|}{\tau_{RC}}\right)$$

Upon setting $R(\tau_o) = 1/e$ and solving the resulting equation numerically, one obtains

$$\tau_{RC} = \frac{\tau_o}{2.146193} \qquad (5.2)$$

The f^{-4} spectrum falls off less rapidly at high Doppler frequencies than does a Gaussian spectrum. The greater high frequency content of random amplitude and phase modulation makes the f^{-4} spectrum a more stressing case for most systems. Demodulation and tracking functions tend to degrade more rapidly in the presence of the "noisier" signal fluctuations corresponding to this power-law spectrum. The fact that the f^{-4} spectrum represents a reasonable worst case, coupled with its ease of implementation in hardware and software, has made it the standard for use in transionospheric flat fading channel simulators.

5.1.3 Flat Fading Channel Simulator.

A fading channel can be represented by a time-varying linear filter. Thus, the received signal is given by the convolution of the transmitted signal and the time-varying channel impulse response function, $h(t,\tau)$. In terms of its complex envelope $u(t)$, the received signal can therefore be written as

$$r(t) = \mathrm{Re}\{u(t)e^{j\omega t}\}$$
$$u(t) = \int_0^\infty m(t-\tau)\,h(t,\tau)\,d\tau \qquad (5.3)$$

where $m(t)$ is the transmitted data modulation, and ω is the carrier angular frequency.

In flat fading, the channel transfer function $H(t,\omega)$ is independent of frequency ω over the signal bandwidth, and the channel impulse response function becomes a Dirac delta function in delay:

$$h(t,\tau) = A(t)e^{j\theta(t)}\,\delta(\tau)\,, \qquad \text{flat fading}$$

It is readily seen that the convolution in Equation 5.3 reduces to multiplication of the transmitted data modulation $m(t)$ times the complex channel modulation $A(t)\exp[j\theta(t)]$. This yields the expression for the received signal given in Equation 5.1. When the data modulation waveform $m(t)$ is real, Equation 5.1 can be rewritten as

$$r(t) = A(t)m(t)\cos\theta(t)\cos(\omega t) - A(t)m(t)\sin\theta(t)\sin(\omega t) \qquad (5.4)$$

Implicit in this representation is the assumption that both the data modulation and the fading spectrum are narrowband with respect to the carrier frequency. Equation 5.4 can be rewritten in terms of two quadrature modulating waveforms:

$$r(t) = I(t)\,m(t)\cos(\omega t) + Q(t)\,m(t)\cos(\omega t + \pi/2)$$

where

$$I(t) = A(t)\cos\theta(t)$$
$$Q(t) = A(t)\sin\theta(t)$$

These equations show that the flat fading received signal can be generated by passing the transmitted signal through a power splitter, shifting the phase of one of the outputs by 90 degrees, and passing the signal and its phase-shifted version through two product modulators. The quadrature modulating waveforms, $I(t)$ and $Q(t)$, represent the fading channel. These waveforms can have any statistical description. Therefore, the simulator is not limited to any particular type of channel as long as it can be adequately represented as narrowband and nonselective over the signal bandwidth.

The f^{-4} flat fading channel is easily synthesized in hardware and software by passing wideband ("white") Gaussian noise through two cascaded single-pole RC lowpass filters in each of two quadrature channels. The filter outputs are the modulating waveforms $I(t)$ and $Q(t)$. When the fading is not completely Rayleigh, some amount of the input signal is fed directly to the output of the channel simulator to represent the specular component, thereby producing Rician statistics.

The flat fading channel model in COMLNK utilizes digital filter techniques to implement the channel impulse response in software. In a digital implementation, the two white Gaussian noise sources are simply two independent sequences from a Gaussian random number generator. Such sequences are easily generated using uniformly distributed random numbers:

$$g_1 = \sqrt{-2\sigma^2 \ln(u_1)}\, \cos(2\pi u_2)$$
$$g_2 = \sqrt{-2\sigma^2 \ln(u_1)}\, \sin(2\pi u_2)$$

where u_1 and u_2 are random numbers drawn from a uniform distribution over the interval (0,1), and g_1 and g_2 are independent zero-mean Gaussian random numbers, each with variance σ^2.

Each single-pole RC filter is implemented digitally using the recursion relation

$$y_k = a\, y_{k-1} + b\, x_k$$

where y_k is the current output of the filter, x_k is the current input, and k is a sampling index. The quantities a and b are filter coefficients, where a is determined by the filter time constant and b is determined by the filter gain. When the gain is set to unity, the values of a and b are given by

$$a = \exp\left(-\frac{\Delta t}{\tau_{RC}}\right)$$
$$b = \left[\frac{(1-a^2)^3}{(1+a^2)}\right]^{1/4}$$

where τ_{RC} is the RC filter time constant (Eq. 5.2), and Δt is the sampling time interval. As in any digital filter implementation, the sampling rate must be rapid enough to minimize spectral aliasing. Because of the stochastic nature of the fading channel, there is no precise definition of the required sampling rate. The sampling rate should be high enough to ensure that the aliased portion of the fading spectrum contains an insignificant fraction of the total power. We recommend choosing Δt in the range $\tau_o/20$ to $\tau_o/40$ to yield 20 to 40 samples per decorrelation time.

If the filter coefficients are held constant, the statistics of the flat fading channel will be stationary. Simulation of nonstationary channels is readily accomplished by varying the filter coefficients. The coefficients are usually adjusted sufficiently slowly that the channel is quasi-stationary insofar as the sampling process is concerned.

The channel simulator described thus far represents only the scattering effects of the flat fading channel. Effects of mean TEC are incorporated by introducing additional phase and time delay variations.

In COMLNK, a full range of channel conditions can be specified for each channel in a link layout. Fading intensity is specified by the S_4 scintillation index, which ranges from zero (nonfading) to unity (Rayleigh fading). The fading rate is specified by the scintillation decorrelation time, τ_o. For dispersive channels, the degree of frequency selectivity is specified by the frequency selective bandwidth, f_o.

The flat fading channel model is implemented in subroutines CHANL1 and CHANL2. All channel models in COMLNK incorporate a continuous phase reconstruction technique to enable the channel phase to be combined with other sources of phase modulation, such as vehicle dynamics and data modulation.

5.2 FREQUENCY SELECTIVE CHANNELS.

When the signal modulation bandwidth is comparable to or larger than the channel coherence bandwidth, frequency selective effects become important. No longer can the effect of the channel be represented by a multiplicative factor on the transmitted signal. Instead, we must return to the time-varying linear filter representation of the channel (Eq. 5.3), wherein the received signal *r(t)* is given by the convolution of the transmitted signal and the channel impulse response function.

The primary measure of the degree of frequency selectivity is the frequency selective bandwidth, denoted by f_o. This is a parameter involved in the second-order signal statistics. The relationship between f_o and the standard deviation of signal time delay jitter σ_t arising from angular scatter is

$$f_o = \frac{1}{2\pi\sigma_t}$$

The value of f_o is commensurate with the maximum modulation rate that the channel will support with little intersymbol interference. Small values of f_o correspond to severe frequency selectivity, while large values are associated with flat fading. Depending on the nature of the propagation medium, the frequency selective bandwidth can vary over a wide range.

Another measure of frequency selectivity is the multipath delay spread of the signal. The delay spread is related to the angular scatter-induced time delay jitter σ_t, and hence is inversely related to the frequency selective bandwidth. Therefore, small values of f_o correspond to large values of multipath delay spread.

5.2.1 Generalized Signal Power Spectrum.

Determination of the statistical properties of a frequency selective propagation channel encompasses the relationship between the statistics of the scattering region and those of the received signal. Much research has been carried out in analytically establishing this link between the propagation environment and the resulting signal statistics.

As shown in Equation 5.3, the frequency selective propagation channel is conveniently represented in terms of the time-varying channel impulse response function $h(t, \tau)$. An equivalent representation may be cast in terms of the Fourier dual, the time-varying channel transfer function $H(t, \omega)$. The former is the channel response at time t to an impulse applied at $t\text{-}\tau$. The latter is the channel response to a sinusoidal excitation at angular frequency ω.

Propagation theory can be used to compute specific realizations of the channel transfer function, but these calculations must be performed numerically using the multiple phase screen technique (Knepp, 1983). What can be obtained analytically is the statistical description of the channel response. This relationship is formulated in terms of the generalized power spectral density (GPSD) of the received signal. The GPSD has been investigated by several researchers, notable among them being (Dana, 1991), upon whose work this subsection is largely based.

The derivation of the GPSD starts with Maxwell's equations, from which the parabolic wave equation is derived. A necessary condition for the parabolic wave equation to be valid is that the phase perturbation over a distance comparable to a wavelength be small compared to one radian. A sufficient condition is that the angular deviation of the wave relative to the principal propagation path be small compared to one radian. These conditions are usually satisfied whenever the propagating wave is not strongly attenuated by the scattering medium.

The parabolic wave equation can be solved to give the received electric field for a specific distribution of the index of refraction. The difficulty lies in the fact that the index of refraction is a random process, so the received electric field is also random. The parabolic wave equation is therefore used to derive an equation for the two-position, two-frequency, two-time mutual coherence function of the electric field.

This derivation would not be possible without the Markov approximation, which is valid under the same conditions that apply to the parabolic wave equation. The Markov approximation assumes that the index of refraction fluctuations are statistically stationary and delta-correlated along the propagation direction. The results then depend on the integrated phase variance along the path.

The solution of the differential equation for the mutual coherence function provides a description of the second-order statistics of the received electric field. The Fourier transform of the mutual coherence function is the GPSD of the received signal. Two of the parameters involved in the GPSD are the decorrelation time, τ_o, and frequency selective bandwidth, f_o.

The GPSD can be written as a product of a Doppler term $S_D(\omega_D)$ and an angle-delay term $S_{K\tau}(K_x, K_y, \tau)$:

$$S(K_x, K_y, \tau, \omega_D) = S_D(\omega_D) S_{k\tau}(K_x, K_y, \tau) \tag{5.5}$$

where ω_D is the Doppler radian frequency, τ is the time delay, and K_x, K_y are related to the scattering angles θ_x, θ_y about the x and y axes, which are located in the receiver plane normal to the propagation path:

$$K_x = \frac{2\pi \sin\theta_x}{\lambda}$$

$$K_y = \frac{2\pi \sin\theta_y}{\lambda}$$

where λ is the RF wavelength. The angle-delay part of the GPSD is

$$S_{K\tau}(K_x, K_y, \tau) = \sqrt{\frac{\pi}{2}}\, l_x l_y \alpha \omega_{coh} \exp\left[-\frac{K_x^2 l_x^2 + K_y^2 l_y^2}{4}\right] \exp\left\{-\frac{\alpha^2}{2}\left[\omega_{coh}\tau - \frac{\Lambda_y (K_x^2 + K_y^2) l_x^2}{4}\right]^2\right\} \tag{5.6}$$

where l_x, l_y are the decorrelation distances of the electric field in the receiver plane[1], Λ_y is an asymmetry factor which is a function of these distances, and ω_{coh} is related to the frequency selective bandwidth f_o:

$$\Lambda_y = \sqrt{\frac{2}{1 + (l_x / l_y)^4}}$$

$$\omega_{coh} = 2\pi f_o \sqrt{1 + \alpha^{-2}}$$

The quantity α is a delay parameter related to the ratio of the time delay spread caused by diffractive effects to that caused by group delay variations along the principal propagation path. In strong scattering where diffractive effects dominate, the value of α is large, and ω_{coh} is then well approximated by $2\pi f_o$.

[1] The decorrelation distances are defined at the $1/e$ points on the two-dimensional spatial autocorrelation function of the complex electric field in the receiver plane. The x-y coordinate system orientation is chosen such that $l_x \leq l_y$. The factor $\sqrt{\pi/2}$ in Equation 5.6 depends on the definitions used for the Fourier transform pairs.

The Doppler part of the GPSD takes on different forms depending on the nature of the propagation medium. The general form is given by (Dana, 1991). Two limiting forms are of particular interest: the frozen-in case and the turbulent case.

In the frozen-in case, the irregularities in the propagation medium act as a cohesive structure that moves relative to the propagation path. The time variations of the received signal are then a straightforward mapping of the electric field diffraction pattern as it drifts past the receiver antenna. In this case there is a strong coupling between the spatial and temporal variations of the received signal. With the drift velocity aligned with the x-direction in the receiver plane, the Doppler component of the GPSD for the frozen-in case is

$$S_D(\omega_D) = 2\pi\tau_o\,\delta(\tau_o\omega_D - K_x l_x) \qquad , \qquad \text{frozen-in case}$$

where $\delta(x)$ is the Dirac delta function. Orientation of the drift velocity in the x-direction minimizes the value of the decorrelation time τ_o. In general, the decorrelation time in the frozen-in case is the time required for the diffraction pattern to drift one decorrelation distance in the receiver plane.

In the turbulent case, irregularities in the propagation medium move differently at different points along the path, causing time variations of the diffracted electric field to be decorrelated at different points in the receiver plane. No coupling remains between position and time or, equivalently, between scattering angle and Doppler frequency. The Doppler spectrum is thus independent of delay; temporal variations of the received signal exhibit the same rate at all delays. The dependence of the Doppler spectrum on Doppler frequency has a Gaussian form:

$$S_D(\omega_D) = \sqrt{\pi}\,\tau_o \exp\left(-\frac{\tau_o^2\omega_D^2}{4}\right) \qquad , \qquad \text{turbulent case}$$

An important signal parameter for systems that utilize large aperture receiving antennas is the angular scattering variance. When the irregularity structure in the propagation medium is anisotropic, as is typically the case in the ionosphere, the scattering angles will be different along the x and y axes in the receiver plane. The standard deviations of the angle-of-arrival fluctuations about the x and y axes are

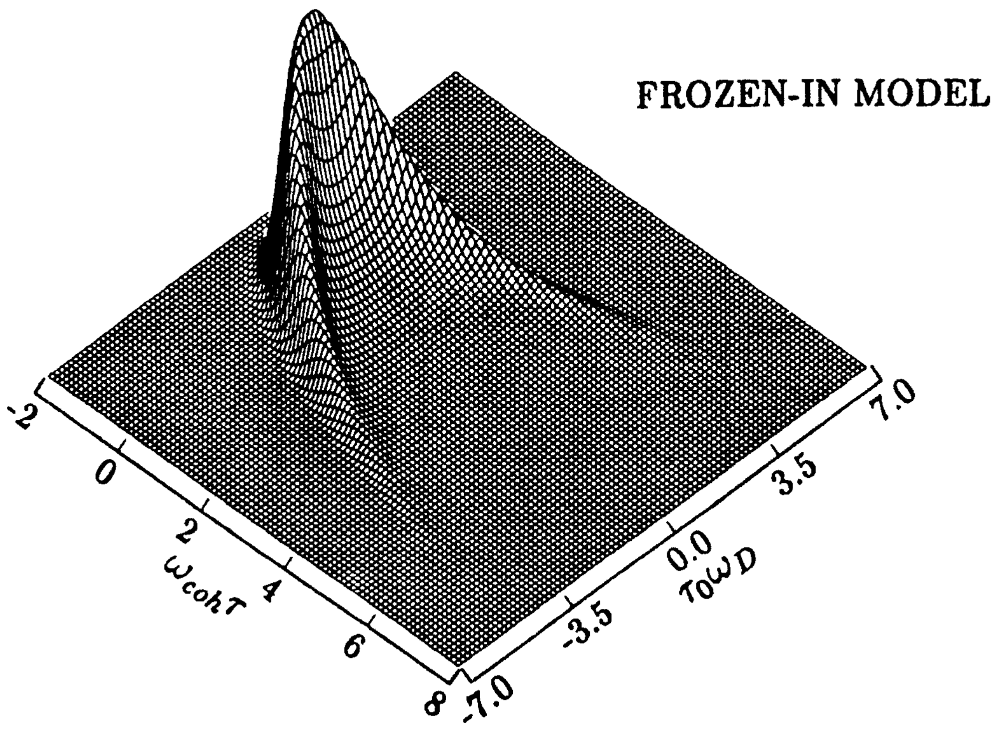
$$\sigma_{\theta_x} = \frac{\lambda}{\sqrt{2}\pi l_x}$$

$$\sigma_{\theta_y} = \frac{\lambda}{\sqrt{2}\pi l_y}$$

If the irregularity structure in the propagation medium exhibits the same scale size irrespective of direction in the plane normal to the propagation path, the decorrelation distances l_x and l_y will be the same. In this isotropic case, the scattering angles are the same in each direction.

Three-dimensional plots of the isotropic scattering function are presented in Figure 5-2 for the frozen-in and turbulent cases. These plots show the mean received power of the scattered signal as a function of normalized Doppler frequency, $\tau_o\omega_D$, and normalized time delay, $\omega_{coh}\tau$. The vertical axis is linear in power with arbitrary units. The delay-Doppler scattering function is obtained by integrating the GPSD over K_x, K_y. The value of the delay parameter α is taken to be four in these examples, so that ω_{coh} is essentially equal to $2\pi f_o$.

The wing-like structure in the upper plot in Figure 5-2 shows that, in the frozen-in case, there is a strong delay-Doppler correlation. This coupling causes signal power arriving at long delays to



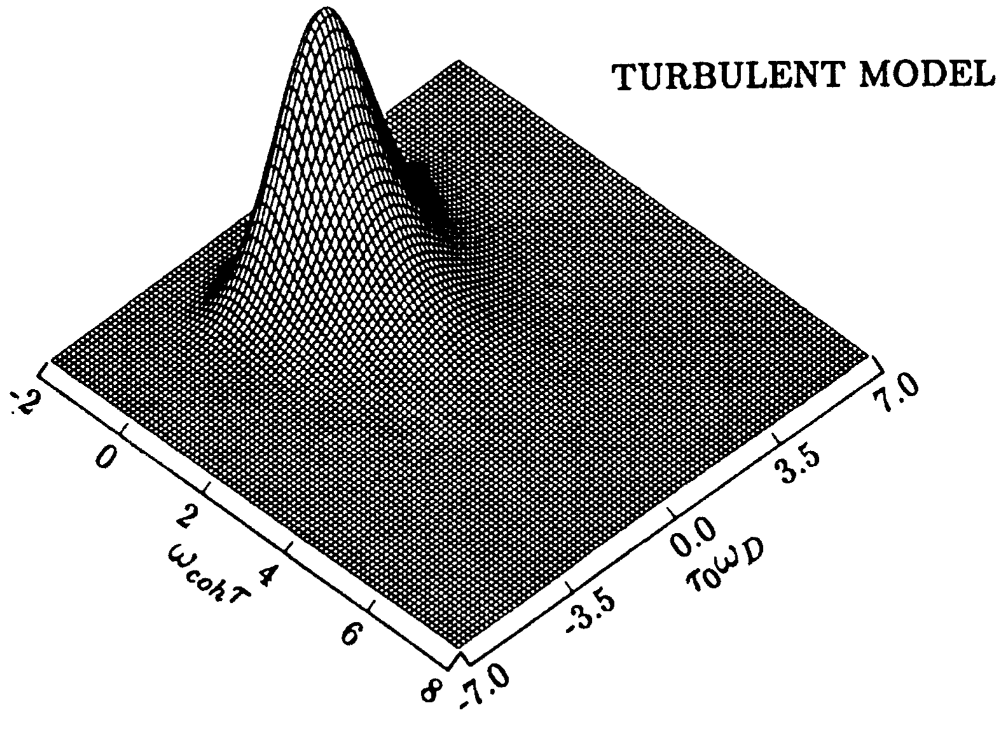


Figure 5-2. Scattering functions for frozen-in and turbulent cases.

have higher Doppler frequency components than that arriving at short delays. Hence, in this model, effective values of the decorrelation time τ_o are smaller when measurements are made at larger time delays relative to the peak signal time-of-arrival.

The lower plot in this figure shows that there is no delay-Doppler coupling in the turbulent case. Both cases have exactly the same power density at each delay. The difference in appearance results from the fact that the turbulent model signal at long delay is more spread out in Doppler frequency and is therefore less obvious in the figure. In the turbulent case, the Doppler spectrum and hence the value of τ_o are independent of delay.

The difference in delay-Doppler coupling between the frozen-in and turbulent cases may affect the performance of systems that employ large aperture receiving antennas. The power arriving at long delays is that which arrives at large angles and consequently is more likely to be attenuated by the narrow beamwidth of a large antenna. In the frozen-in case, this antenna spatial filtering increases the effective value of τ_o at the antenna output compared to that at the face of the antenna. In the turbulent case, because the Doppler spectrum is independent of delay there is no spatial filtering of τ_o.

In any case, antenna spatial filtering gives rise to scattering loss when scattering angles are comparable to or larger than the antenna beamwidth. An equivalent statement is that spatial decorrelation of the received electric field across the antenna aperture causes a reduction in the effective gain of the antenna. An antenna is nominally a spatial matched filter for an incident plane wave, and random spatial variations of the scattered wave cause a loss due to mismatch. The magnitude of this loss is a function of the ratio of antenna aperture diameter to decorrelation distance of the incident wave. Scattering loss also depends on the antenna pointing angle relative to the mean angle-of-arrival of the scattered signal.

If the antenna is pointed directly along the mean signal angle-of-arrival, the energy that is attenuated by a narrowbeam antenna is that arriving at large scattering angles and hence at relatively large time delays. The effect of a large antenna then is to reduce the delay spread in the signal at the antenna output, which increases the effective value of f_o. The resulting reduction in intersymbol interference in a high data rate communications receiver is a potential beneficial effect of the antenna spatial filtering of f_o. However, the benefit must be weighed against the concomitant scattering loss.

A thorough analysis and discussion of antenna filtering, including the dependence on pointing angle and other factors, is given by (Dana, 1991). The reader should consult that reference to obtain analytical and numerical results for scattering loss, together with values of filtered frequency selective bandwidth and other related information.

In conjunction with Rayleigh first-order signal statistics, the GPSD provides a complete statistical description of the frequency selective scattering channel. Given values of the signal decorrelation time τ_o and frequency selective bandwidth f_o, the GPSD can be used to generate specific realizations of the channel impulse response function. The channel impulse response then enables specific realizations of the received signal to be generated.

5.2.2 Frequency Selective Effects.

Frequency selective effects are most likely to be encountered in applications utilizing wide signaling bandwidths. An example of such a system is one that uses PN direct-sequence spread spectrum modulation (Bogusch, *et al.*, 1981). A PN spread spectrum waveform is formed by combining a low rate binary data sequence with a high rate binary pseudorandom code sequence. The modulo-two sum of these sequences is used to phase-modulate a carrier. The ratio of code rate to data rate is typically quite large.

In a PN receiver, the signal is despread (*i. e.*, the pseudorandom code sequence is removed) by mixing or correlating the received signal with a locally generated replica of the PN code sequence. If the receiver timing is correctly aligned with the received signal, and if the signal waveform is undistorted, the correlation operation ideally cancels the phase changes introduced by the PN code, leaving only the data phase.

In a frequency selective channel, operation of the PN code correlator may be significantly degraded. Because each spectral component of the wideband signal experiences different amplitude and phase variations, the local reference waveform in the receiver is no longer matched to the received signal. The correlator output exhibits delay spread with multiple peaks.

Relative motion of the propagation path and scattering region causes the waveform distortion to vary with time. The problems faced by a receiver are first to acquire carrier frequency and PN code timing, and then to track the peak signal energy as best possible. The most critical function is usually that of maintaining code lock, *i. e.*, keeping the receiver time estimate near the signal time-of-arrival. Even when code lock is maintained, energy loss due to correlator waveform distortion is generally inevitable in frequency selective channels. This loss degrades the performance of all signal processing and data demodulation functions.

While there are many similarities between a wide bandwidth PN spread spectrum waveform and a high data rate waveform, frequency selective effects are manifested somewhat differently in the two classes of systems (Bogusch, *et al.*, 1983). In a high data rate system, waveform distortion produces intersymbol interference (ISI), wherein energy from preceding symbols interferes with data demodulation.

ISI can cause conventional matched filter demodulation to fail catastrophically. The degradation cannot be mitigated by increasing the signal level, because the problem is signal energy competing with itself, not noise. An increase in the signal level simply produces an increase in the level of interference as well.

Mitigation of ISI involves adaptive equalization techniques. Equalization of a dispersive channel requires a method of gathering the delay spread signal energy together in the proper symbol period. Equalization of a time-varying channel requires that the signal processing adapt to changing channel conditions. Both requirements can be satisfied with adaptive decision-feedback equalization, which can significantly increase channel capacity in the presence of frequency selective scintillation.

5.2.3 Frequency Selective Channel Simulator.

Simulation of frequency selective channels is straightforward, albeit more complicated than flat fading channel simulation. Two methods of implementing frequency selective channel simulators are in use. These are the Fourier synthesis method (frequency domain) and the convolution synthesis method (time domain). The Fourier synthesis method has received extensive application in previous software simulations. The convolution synthesis method has received extensive application in hardware channel simulators, and is the method employed in COMLNK.

The basis for the Fourier synthesis technique is provided by the Fourier dual of the convolution integral in Equation 5.3. Here the equation is rewritten in terms of the signal spectrum $M(\omega)$, given by the Fourier transform of the modulation waveform $m(t)$, and the time-varying channel transfer function $H(t, \omega)$, given by the Fourier transform of the channel impulse response $h(t, \tau)$:

$$\begin{aligned} r(t) &= \mathrm{Re}\{u(t)e^{j\omega t}\} \\ &= \mathrm{Re}\{u(t)\}\cos(\omega t) - \mathrm{Im}\{u(t)\}\sin(\omega t) \end{aligned} \tag{5.7}$$

$$u(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} M(\omega)H(t,\omega)e^{j\omega t}\,d\omega$$

Given values of the signal spectrum and channel transfer function at a set of discrete frequencies within the signal bandwidth, the complex envelope *u(t)* can be computed using a discrete Fourier transform. Equation 5.7 shows that the received signal then can be generated using a vector modulator, with the I and Q modulating waveforms given by Re{*u(t)*} and Im{*u(t)*}, respectively.

The convolution synthesis method of simulating a frequency selective channel is the time domain dual of the Fourier synthesis method. The convolution integral, Equation 5.3, is applied directly. Both methods of synthesizing a frequency selective signal therefore require an integration to be performed at each time that the signal is to be sampled. The Fourier synthesis method implements the integration (Eq. 5.7) using a discrete Fourier transform. The convolution synthesis method implements the integration (Eq. 5.3) using a transversal filter.

Perhaps the primary difference in the two channel simulation methods is that in convolution synthesis the transmitted modulation waveform *m(t)* is input directly rather than in spectral form. This is very convenient in hardware and software implementations, and is the reason for using convolution synthesis in COMLNK.

5.2.4 Tapped Delay Line Channel Model.

Simulation of the frequency selective channel impulse response is readily accomplished using a tapped delay line channel model. A hardware implementation of this model is employed in the DNA NELS II channel simulator (Cross, 1987). A software implementation is employed in COMLNK. Detailed discussion of the theory and implementation of the tapped delay line model is available in the literature (Dana, 1991; Bogusch and Michelet, 1993; Dana, 1994).

The basis of the tapped delay line model is the convolution integral, Equation 5.3. A discrete version of this integral is implemented using a tapped delay line transversal filter. The output of each tap is weighted by the complex amplitude of the channel impulse response function evaluated at the tap delay. The transmitted signal is fed into the tapped delay line, and the sum of the weighted tap outputs provides the received distorted signal.

A discrete version of Equation 5.3, suitable for implementation in a tapped delay line transversal filter, is given by

$$u(t) = \sum_{k=0}^{K-1} m(t-k\Delta\tau)\,h(t,k\Delta\tau)\,\Delta\tau \tag{5.8}$$

where K is the number of taps on the delay line, $\Delta\tau$ is the delay spacing between each tap, and the complex tap weights $h(t,k\Delta\tau)$ are given by the sampled values of the channel impulse response function at the delays $k\Delta\tau$.

Generation of the complex tap weights is done using the GPSD. Under strong scattering conditions where the GPSD is valid, the real and imaginary parts of the channel impulse response function are independent, zero-mean, Gaussian random variables. A key assumption involved in generating realizations of the channel impulse response function is that the channel is wide-sense statistically stationary in space, frequency, and time. The impulse response function is then delta-correlated in angle, delay, and Doppler frequency. This allows the impulse response function to be generated from white Gaussian noise in the angle, delay, and Doppler frequency domains, and then Fourier transformed to the space and time domains.

A detailed discussion of the general approach to generate specific realizations of $h(t, \tau)$ using the GPSD is provided by (Dana, 1991). Realizations of the channel impulse response are readily generated for both the frozen-in and turbulent cases, as well as for intermediate cases.

A particularly simple special case arises when one considers the turbulent model with isotropic scattering and diffractive effects dominant (delay parameter α very large). In this case, the GPSD (Eq. 5.5) takes the form

$$S(\omega_D, \tau) = S_D(\omega_D) S_\tau(\tau)$$

and the delay term (Eq. 5.6) reduces to

$$\begin{aligned} S_\tau(\tau) &= 2\pi f_o \exp(-2\pi f_o \tau) \quad , \quad \tau \geq 0 \\ &= 0 \quad , \quad \tau < 0 \end{aligned} \tag{5.9}$$

Specific realizations of the channel impulse response function are generated by choosing a discrete set of delay taps defined by

$$\tau_k = k\Delta\tau \quad , \quad k = 0, 1, 2, ..., \text{K-1}$$

The tap spacing $\Delta\tau$ and the number of taps K are chosen to represent essentially all of the signal energy. The total delay spread depends on the signal bandwidth, determined by the modulation symbol period T, and the channel bandwidth, defined by the frequency selective bandwidth f_o. The delay spacing and number of taps needed to represent the channel are therefore functions of these parameters. If P_τ denotes the fraction of the total signal power that is to be contained in the delay grid, then integration of Equation 5.9 gives the following expression for the required number of delay taps:

$$\text{K} = 1 - \frac{\ln(1 - P_\tau)}{2\pi f_o \Delta\tau}$$

There should be at least two delay taps per modulation symbol period to represent the modulation frequency spectrum. Therefore, the channel tap spacing in COMLNK is set at one-half the modulation symbol period. Recent work (Dana, Milner, and Bogusch, 1995; Dana and Bogusch, 1998) shows that a better representation of the channel is obtained with a smaller tap spacing. Hence, the tap spacing may be reduced to one-fourth the symbol period (or become a user input) in a future release.

In any case, the number of taps is chosen using the specified value of frequency selective bandwidth to encompass at least 97.5 percent of the delay-spread signal energy. Therefore, with $\Delta\tau$ = T/2 and P_τ = 0.975, the number of taps is

$$\text{K} \geq 1 + \frac{3.7}{\pi f_o \text{T}}$$

The mean signal energy at each tap is obtained by integrating Equation 5.9 over the corresponding delay interval:

$$P_k = \exp(-2\pi f_o \tau_k) - \exp(-2\pi f_o \tau_{k+1})$$

The sampled channel impulse response function for the k-th delay tap is then

$$h(t, k\Delta\tau)\Delta\tau = (x_k + jy_k)\sqrt{P_k} \tag{5.10}$$

where x_k and y_k are independent, zero-mean Gaussian random numbers with equal variances of 0.5. This complex Gaussian factor provides the Rayleigh fading time history for each delay sample. It is important to remember that the channel impulse response is delta-correlated in delay, and hence x_k, y_k are independent for different values of k corresponding to different delay taps.

The Doppler spectrum of the fading process is determined by the power spectrum of the complex Gaussian factor in Equation 5.10. Digital filter techniques are used to generate the complex samples of $x_k + jy_k$ by filtering white Gaussian noise. To facilitate implementation of time-domain filters, it is convenient to assume that the Doppler spectrum can be represented by a power-law form with an even spectral index.

The f^{-4} spectrum described earlier could be used here, but this spectrum does not fall off rapidly enough with Doppler frequency for use in a frequency selective channel. Thus, we adopt an f^{-6} Doppler spectrum (Dana, 1994). The f^{-6} frequency selective channel is easily implemented in hardware and software by passing wideband Gaussian noise through three cascaded single-pole RC lowpass filters in each of two quadrature channels for each tap in the delay line. The filter coefficients are established by the values of scintillation index and decorrelation time. The filter outputs provide the complex tap weights in the transversal filter.

Separate filters with independent noise sources are used for each delay tap. The resulting samples of the channel impulse response from Equation 5.10 are used directly in the tapped delay line model of Equation 5.8. Hence, each tap weight is an independent complex Gaussian process having an f^{-6} Doppler spectrum with the specified decorrelation time.

In COMLNK, a full range of channel conditions can be specified for each channel in a link layout. Fading intensity is specified by the S_4 scintillation index. For generality, the frequency selective channel model can encompass the range from weak to strong scattering. The fading rate is specified by the decorrelation time, τ_o. The degree of frequency selectivity is specified by the frequency selective bandwidth, f_o. The frequency selective channel model is implemented in subroutine CHANL3.

5.3 EFFECTS OF TOTAL ELECTRON CONTENT.

For transionospheric propagation paths, a complete description of the channel involves the deterministic effects of the mean TEC, as well as the random scattering effects of ionization irregularity structure. It is particularly important to include TEC effects when investigating signal acquisition. The TEC increases the signal time delay along the propagation path, advances the signal phase, and may increase the apparent mean Doppler frequency shift. All of these effects need to be considered in the design of algorithms for acquisition of signal timing, phase, and frequency.

Consider an electromagnetic wave with angular frequency ω propagating through the ionosphere in the positive z direction. Each spectral component of the wave is governed by an equation of the form

$$E(z,t) = E_o \exp\left[j\left(\omega t - \int k(z)\,dz\right)\right] \tag{5.11}$$

where the propagation coefficient $k(z)$ in a collisionless plasma is

$$k(z) = \frac{\omega}{c}\sqrt{1 - \frac{\omega_p^2}{\omega^2}}$$

The plasma frequency, ω_p, is given by

$$\omega_p^2 = 4\pi c^2 r_e \, n_e(t,z)$$

where r_e is the classical electron radius (2.817939 x 10^{-15} m), c is the speed of light, and $n_e(t, z)$ is the electron number density per cubic meter, which may vary with time.

From Equation 5.11, it is seen that the transfer function of a smooth ionosphere with negligible absorption is

$$H(t,\omega) = e^{j\theta(t,\omega)}$$

where the phase function $\theta(t,\omega)$ is obtained by integrating the propagation coefficient $k(z)$ along the path from the transmitter located at $z = 0$ to the receiver located at $z = R$:

$$\begin{aligned}\theta(t,\omega) &= -\int_0^R k(z)\,dz \\ &= -\frac{\omega}{c}\int_0^R \sqrt{1 - \frac{\omega_p^2}{\omega^2}}\,dz\end{aligned} \qquad (5.12)$$

The phase function can be expanded in a Taylor series about the carrier frequency. Retaining terms through third-order, the transfer function of a smooth collisionless ionosphere can be written as

$$H(t,\omega) = \exp\left[j\left(\theta_0 + \theta_1 \nu + \theta_2 \frac{\nu^2}{2} + \theta_3 \frac{\nu^3}{6} \right)\right]$$

where ν is the frequency relative to the carrier frequency. The Taylor series expansion coefficients are given by derivatives of Equation 5.12 with respect to frequency, evaluated at the carrier frequency. When the plasma frequency is much less than the carrier frequency, the resulting coefficients can be approximated as follows:

$$\theta_0 = 2\pi c\, r_e \frac{N_T}{\omega} - \omega \frac{R}{c} \qquad (5.13)$$

$$\theta_1 = -2\pi c\, r_e \frac{N_T}{\omega^2} - \frac{R}{c} \qquad (5.14)$$

$$\theta_2 = 4\pi c\, r_e \frac{N_T}{\omega^3} \qquad (5.15)$$

$$\theta_3 = -12\pi c\, r_e \frac{N_T}{\omega^4} \qquad (5.16)$$

where the range R is measured in meters, and N_T is the TEC in electrons per square meter:

$$N_T = \int_0^R n_e(t,z)\,dz$$

The quantity θ_0 is the mean signal phase shift due to range and TEC, and θ_1 is the total mean propagation time delay with the sign reversed. The remaining two phase factors are the first two dispersive terms due to mean TEC. In general, all four phase factors are functions of time because of TEC variations and path motion.

The received carrier frequency exhibits a mean Doppler frequency shift if either the range or the TEC varies with time. The Doppler shift is given by the first time derivative of Equation 5.13, divided by 2π to obtain units of Hertz:

$$f_D = \frac{r_e \lambda}{2\pi}\frac{dN_T}{dt} - \frac{1}{\lambda}\frac{dR}{dt} \tag{5.17}$$

where λ is the RF wavelength at the carrier frequency. By replacing the first time derivatives in this equation with higher-order derivatives, one obtains equations for rate of change of frequency due to range or TEC acceleration (second derivative), jerk (third derivative), and yank (fourth derivative).

Rewriting Equation 5.14 in terms of the wavelength and changing the sign gives the following equation for the mean propagation time delay in seconds:

$$t_d = \frac{r_e \lambda^2 N_T}{2\pi c} + \frac{R}{c} \tag{5.18}$$

Increasing range (positive *dR/dt*) corresponds to the transmitter and receiver moving away from each other, which yields a negative Doppler shift as shown in Equation 5.17. As the range increases, so does the signal time delay. Note, however, that increasing TEC (positive dN_T/dt) gives rise to a positive Doppler shift while causing time delay to increase. In other words, there is a sign reversal in the relationship between delay and Doppler effects of range and TEC.

This sign reversal between range and TEC effects can have a significant impact on the design and performance of receivers that use carrier Doppler estimates to aid signal time delay acquisition and tracking. If conventional Doppler aiding is applied on the assumption of range variation, the "aiding" will be in the wrong direction if the Doppler shift is actually caused by TEC variation. Whether or not this is a serious problem depends on the level of TEC, carrier frequency, and modulation chip period.

Range and TEC Dynamics.

Design and evaluation of tracking loops requires some method of simulating the signal phase and time delay dynamics that result from motion of the transmitter and receiver platforms, or from variations in ionospheric TEC, or both. In fading channels, it is particularly important to provide the loops with something to track. In most cases, the loops suffer the greatest stress when the signal happens to suffer a deep fade at the same time that carrier dynamics are significant.

Three generic dynamics profiles are incorporated in COMLNK to exercise the demodulator tracking loops. These profiles are referred to as the initial-value profile, the sinusoidal profile, and the square-wave profile. Each of these profiles can be used to represent variations in range or TEC along the signal propagation path. In each case, values can be specified for the quantity and its first four time derivatives.

For example, the initial value profile can be used to impose an initial range rate to provide an initial carrier Doppler offset, which then can abruptly drop to zero. Sinusoidal variation of range or its time derivatives can also be selected with a specified period. The square-wave profile enables a quantity or its derivatives to undergo a balanced square wave with a given period. In all cases, lower derivatives are obtained through integration. By selecting the dynamics profiles and associated range and TEC derivatives, tracking loops can be stressed to determine their break points.

Range and TEC dynamics profiles are implemented in subroutine RTPDYN, which is included in the channel modules.

5.4 TRANSPONDER CHANNELS.

Communications links often involve more than one propagation path. An example is a satellite link, which has an uplink from a ground or airborne transmitter to the satellite, and a downlink to a receiver at another location. There may be more than two paths if, for example, the signal is cross-linked between satellites before being relayed back to the ground.

Application of the preceding channel modeling techniques to the case of two or more propagation paths is straightforward. In general, each path will exhibit different fading characteristics, and the fading will be uncorrelated from one path to another because of large spatial separations. Each path can then be simulated independently using the appropriate channel model, with different values of scintillation index, decorrelation time and frequency selective bandwidth for the different paths.

If the satellite transponder (or intermediate node whatever the type of system) is nonlinear, each propagation path must be individually treated in this manner. This provides the only correct way of analyzing or testing the end-to-end performance of a multiple-path link with nonlinear intermediate nodes. An example of such a system is one utilizing processing satellites that demodulate the uplink signal, perhaps perform some error correction, and then remodulate the data onto the downlink carrier.

5.4.1 Linear Bent-Pipe Transponder.

A relay through a bent-pipe transponder is an example of a situation where a linear repeater may be an adequate model. The transponder could be on board a satellite, or it might be a microwave repeater. Either way, two distinct signal propagation paths are involved, and either or both of them may be fading. Because of physical separation, the two channels will generally exhibit different characteristics.

When the transponder can be represented by a linear repeater, it is possible to analyze the statistics of the combined two-path channel if each individual path exhibits Rayleigh fading. The first-order signal statistics for the two-path channel are not Rayleigh, but instead are given by the product of two independent Rayleigh processes.

Consider first the case of flat Rayleigh fading. The random modulation imposed by each path can be written as $A_i(t)\exp[j\theta_i(t)]$, where the subscript i is a path index. With a linear transponder, the resulting two-path channel modulation is given by the product of two independent factors. Each factor can be written as a complex Gaussian random process. The probability density function of the received power $P=(A_1A_2)^2$ and the corresponding cumulative distribution are (Dana, 1982)

$$f(P)=\frac{2}{P_0}K_0\left(2\sqrt{\frac{P}{P_o}}\right)$$

$$F(P)=1-2\sqrt{\frac{P}{P_0}}K_1\left(2\sqrt{\frac{P}{P_0}}\right)\quad .$$

where P_0 is the mean received power, and $K_0(x)$ and $K_1(x)$ are modified Bessel functions.

When these distributions are compared with the single-path Rayleigh distribution, it is found that the two-path channel has a higher probability of deep fades than a single Rayleigh path. For example, a Rayleigh channel has a 0.01 percent probability of fades below -40 dB, whereas this probability increases to 0.1 percent in the two-path channel.

The S_4 scintillation index also reflects the more severe fading, increasing from unity for a single Rayleigh path to the square root of three for the two-path case with uncorrelated fading on each path (Dana, 1993). If the two paths were perfectly correlated, as in a monostatic radar, the value of S_4 increases further to the square root of five and the mean received power doubles (+3 dB). It is possible to model this monostatic case in COMLNK simply by setting all parameters of both transponder paths equal, including the random number generator seeds.

The effective scintillation decorrelation time for the two-path channel is (Dana, 1982)

$$\tau_{oeff} = \frac{\tau_{o1}\tau_{o2}}{\sqrt{\tau_{o1}^2 + \tau_{o2}^2}} \tag{5.19}$$

where τ_{o1} and τ_{o2} are the decorrelation times of the two individual paths. The decorrelation time of the two-path channel is smaller than that of either of the individual paths. If one of the paths has a much smaller decorrelation time than the other, the effective value of τ_o will approach the smaller value. Equation 5.19 is strictly valid only for a Gaussian fading power spectrum, but it is representative of the behavior that can be expected for other fading spectra as well.

For the case of frequency selective fading, the effective frequency selective bandwidth for the linear transponder two-path channel is given by an equation of the same form:

$$f_{oeff} = \frac{f_{o1}f_{o2}}{\sqrt{f_{o1}^2 + f_{o2}^2}}$$

where f_{o1} and f_{o2} are the frequency selective bandwidths of the two individual paths. As for the decorrelation time, the frequency selective bandwidth of the two-path channel is smaller than that of either of the individual paths. A two-path linear transponder channel is therefore more severe than the single-path Rayleigh fading channel in all respects.

It is not difficult to model a linear two-path frequency selective fading channel using the techniques that have been outlined. Using the convolution synthesis method, one first obtains independent channel impulse response functions $h_1(t, \tau)$ and $h_2(t, \tau)$ for the two individual paths. The impulse response function for the two-path channel is then given by the convolution of the two impulse response functions. The foregoing equations for the effective decorrelation time and frequency selective bandwidth of the two-path channel are not actually used to model the channel, but they illustrate the statistical characteristics of the resulting scintillation.

Application of the preceding channel models to linear transponder links is straightforward. The impulse response of two cascaded channels is obtained by convolution of their individual impulse response functions. The resulting impulse response is then convolved with the transmitted signal to obtain the signal at the receiver.

At each channel sampling time, the linear transponder module performs a discrete convolution of the tapped-delay line impulse response of the two channels. The result is then placed on the output bus. Note that each channel can have a different number of taps in its impulse response; a flat fading channel is represented by a single tap.

Any combination of flat fading and frequency selective fading channels can be simulated by this procedure. If N_1 and N_2 denote the number of taps that represent the individual channels at any time, the number of taps in the convolved impulse response is $N_1 + N_2 - 1$. The two individual channels may be statistically stationary, or either or both of them can be time varying. Consequently, the number of taps in the response functions can vary as a function of time. The user need not be concerned with any of these details, as they are handled automatically by COMLNK. The procedure is implemented in subroutine XPONDR.

5.4.2 Partial and Fully Processing Transponders.

Much more complicated links involving multiple channels can be constructed, as illustrated by many of the example layouts included on the distribution diskettes and illustrated earlier in Section 1. Links involving partial and fully processing transponders are highly nonlinear and therefore are simulated directly by explicitly incorporating each processing function in the layout.

Each channel in a multi-segment layout is simulated using one of the channel models described in this section. Again, the user need not be concerned with the details of the channel models. Once the type of modulation is specified and channel parameters are given for each link segment, COMLNK automatically selects the appropriate channel model for each combination of waveform and signal parameters.

5.5 NONSTATIONARY CHANNELS.

The foregoing channel models provide specific realizations of the random channel impulse response for given values of scintillation index, decorrelation time, and frequency selective bandwidth. If the values of these signal statistical parameters remain constant over time periods of interest, the channels are statistically stationary.

This is not the case in many physical channels, however. For example, a signal propagation path between an aircraft and a satellite may move through regions of naturally occurring ionization structure in the polar or equatorial ionosphere, causing scintillation intensity and fading rate to exhibit significant variation with time. Transionospheric paths in a nuclear environment are also likely to encounter significant changes in propagation conditions. Wireless communications with a moving vehicle in an urban environment is yet another example of time-varying channel statistics.

COMLNK accommodates such applications by including the capability to simulate statistically nonstationary channels in two ways. One way is to vary the statistical parameters in an on-line channel model while generating a specific realization of the channel impulse response function. The other way is to sample the channel impulse response off-line, using either a channel simulator or field measurements of an actual channel, and use the recorded data directly.

5.5.1 Time-Varying On-Line Channels.

When this option is exercised, values of the signal statistical parameters (S_4, τ_o, f_o) along with path attenuation, noise temperature, and TEC are specified at a set of times. These values are provided in a channel parameter file that the program reads during a simulation run. At a user-specified time interval, values of the channel parameters are interpolated from the file data and used to adjust the digital filter coefficients in the on-line channel models.

Time-varying channels can vary through a wide range of conditions, from nonfading AWGN, to flat Rician fading, to flat Rayleigh fading, to frequency selective fading, and back. Any set of channel parameters can be specified in any sequence. At each update time, the time-varying channel routine determines which channel model should be invoked and sets up the state variable dataset for that model. The appropriate channel routine is then linked into the interrupt controller.

One consequence of time-varying channel conditions is that conventional measures of performance, such as cumulative average error rates, become less useful and often quite meaningless. Running average error rates provide far superior performance measures in nonstationary channels. Running averages are provided in an optional plot output file.

The SCENARIO.DAT and SCENARIO.CHN files on the distribution diskettes provide an example of the capability to simulate nonstationary channels using the on-line models. A very wide range of channel conditions can be simulated with this capability, which is implemented in subroutine CHANLT.

5.5.2 Recorded Channels.

Direct utilization of recorded channel response data provides another way of incorporating the effects of channels that may exhibit nonstationary signal statistics. The channel is represented by inphase and quadrature digital samples of the impulse response at a specified sampling rate. A tapped delay line representation is again employed, with a single tap providing a flat channel response and multiple taps providing a frequency selective response.

The sampled channel response is provided in a channel data file that the program reads during a simulation run. At each sampling time, the I and Q data are read and used directly to represent the channel impulse response. The sampling rate and number of delay taps can vary as a function of time in the data file.

The channel data can be obtained from any source, and hence there is no limit on the range of signal statistics that can be accommodated. Not only can the channel be nonstationary, it can have any type of first-order and second-order statistical properties.

The RECORDED.DAT and RECORDED.CIQ files on the distribution diskettes provide an example of the capability to use recorded channel data. A virtually unlimited range of channel conditions can be simulated using this capability, which is implemented in subroutine CHNLIQ.

5.6 RADIO FREQUENCY INTERFERENCE.

A communications system that relies on radio wave propagation is inherently susceptible to interference from other sources of radio frequency energy. From the standpoint of the communications receiver, it matters little whether these sources of interference are considered hostile or friendly. In either case, undesirable energy arrives at the receiver and competes with the desired signal. Consequently, the term radio frequency interference may refer to either intentional or unintentional jamming.

Once the interfering energy is captured by the receiver antenna and passes through the RF and IF stages and is downconverted to baseband, it combines with signal energy and noise to form a composite voltage waveform that is sampled and digitized for further processing. At this point, it has become indistinguishable from signal or noise energy. There is no way that the receiver can discriminate between the various energy sources on a sample-by-sample basis. Rather, any exorcising of jammer energy relies on the processing of batches of samples in an attempt to suppress the effects of interference and identify known signal characteristics that have become immersed in a sea of interference and noise.

Accurate simulation of jamming or RFI therefore relies on development of detailed mathematical representations of the inphase and quadrature baseband voltage components produced by the interfering waveforms. Three basic types of jamming waveforms are considered here: noise, a comb of tones, and linear frequency modulation (FM). Each of these waveforms may be pulsed or continuous in time (although linear FM implies a finite sweep period), and the spectral occupancy of each can vary greatly with respect to the communications system bandwidth.

5.6.1 Noise Jamming.

When the interference exhibits a power spectrum that is essentially flat over the receiver sampling bandwidth, it can be represented mathematically in the same manner as Gaussian noise. Thus when noise jamming is encountered, the jamming components of the I and Q samples can be represented by additional independent zero-mean Gaussian random variables with equal variances. The variance of each of the I and Q noise jamming components can be written as

$$\sigma_J^2 = \frac{N_J f_s}{2} \tag{5.20}$$

where N_J is the effective one-sided noise spectral density of the received interference, and f_s is the receiver A/D sampling rate. The mean noise jamming power in the A/D sampling bandwidth is therefore equal to $2\sigma_J^2$ or $N_J f_s$.

Given the ratio J/S of received jammer power to received signal power, together with the mean received signal-to-noise ratio in the A/D sampling bandwidth $\mathrm{C}/(\mathrm{N_o} f_s)$, the total jammer power J is given by

$$J = \left(\frac{\mathrm{J}}{\mathrm{S}}\right)\left(\frac{\mathrm{C}}{\mathrm{N_o} f_s}\right)(\mathrm{N_o} f_s) \tag{5.21}$$

where $\mathrm{N_o}$ is the receiver noise spectral density. Note that the signal power S in the ratio J/S is the same as the mean received carrier power C in the ratio $\mathrm{C/N_o}$. With reference to Equations 4.2 and 4.3, the mean noise power in the A/D samples, $\mathrm{N_o} f_s$, is related to specified receiver design values as follows:

$$\mathrm{N_o} f_s = 2\left(\frac{n_o}{G_o}\right)^2$$

where n_o is the design value of the one-sigma noise quanta in the A/D samples, and G_o is the design value of the A/D converter voltage gain control word.

Once the value of the total jammer power J is computed from the given values of J/S and $\mathrm{C/N_o}$ using Equation 5.21, the noise jammer spectral density is given by

$$N_J = \frac{J}{B_J}$$

where B_J is the bandwidth in which the total noise jammer power is measured and over which that power is evenly spread. The jammer bandwidth B_J is defined as the product of the full bandwidth utilized by the communications system times the jammer fractional bandwidth:

$$B_J = F_J B_{sys}$$

where B_{sys} is the full system bandwidth, and F_J is the jammer bandwidth ratio for partial-band jamming.

For frequency-hopped systems, the system bandwidth is defined as the frequency-hop bandwidth, and the jammer bandwidth ratio F_J may range from zero to unity. This is the only case in which partial-band noise jamming is considered. When partial-band noise jamming is specified, only those hop frequencies that fall within the specified jammer frequency band encounter jamming interference in the I and Q sampling process.

For systems that do not employ frequency hopping, the system bandwidth is defined to be the signal bandwidth or the A/D sampling bandwidth, whichever is larger. In direct-sequence spread-spectrum systems, the system bandwidth is equal to the PN code chip rate. In these non-hopped cases, the noise jammer power is measured in the full system bandwidth and is considered to be uniformly spread over that bandwidth, resulting in full-band noise jamming.

In all cases, when the signal carrier frequency falls within the bandwidth of the noise jammer, random samples of the jamming voltage are computed and added to the signal and noise voltages in the I and Q baseband channels. The jammer noise voltage samples are drawn from a zero-mean Gaussian distribution with a variance given by Equation 5.20. If the propagation channel between the jammer and receiver exhibits scintillation fading, the variance of the jammer I and Q samples is multiplied by the power response of the fading channel.

A useful measure of jammer effectiveness is the noise equivalent value of the ratio of channel bit energy to noise density plus interference density, denoted here as $E_{cb}/(N_o+N_I)$. To relate this ratio to the given jammer parameters, first rewrite it as follows:

$$\frac{E_{cb}}{N_o + N_I} = \frac{1}{N_o/E_{cb} + N_I/E_{cb}}$$

The interference density N_I is equal to the total jamming power J divided by the total bandwidth utilized by the system, B_{sys}. The channel bit energy E_{cb} is equal to the signal power S divided by the channel bit rate R_{cb}. When these relationships are inserted into the second term of the denominator, one obtains the following expression:

$$\frac{E_{cb}}{N_o + N_J} = \frac{1}{N_o/E_{cb} + (J/S)(R_{cb}/B_{sys})} \tag{5.22}$$

While this noise equivalent value of E_{cb}/N_o is not used in the simulation, it is given as output to provide a figure of merit to assist in evaluating jammer effectiveness. For example, a full-band noise jammer has precisely the same effect on data demodulation performance as that which would result if the signal-to-noise ratio were reduced in accordance with Equation 5.22. Now, if another jammer having the same power but a different waveform or spectral occupancy produces less degradation than would be predicted by this expression, a jammer designer can conclude that full-band noise would be a better utilization of the available jammer power. On the other hand, a system designer can conclude that his signal waveform design has rendered the more sophisticated jammer less effective than noise.

5.6.2 Tone Jamming.

Jamming or RFI may take the form of a set of continuous wave (CW) monochromatic frequencies or tones. Any number of jammer tones may be present in the system bandwidth. Each received tone can be represented mathematically as

$$r(t) = \mathrm{Re}\{A(t)\exp[j\omega t + j\theta(t)]\}$$

where ω is the tone angular frequency, $A(t)$ is the tone amplitude including any amplitude fading imposed by the jammer propagation channel, and θ(t) is the tone phase including any jammer Doppler shift and channel phase scintillation.

The total received jammer power J is calculated from the given values of J/S and C/N_o as described above (Eq. 5.21). When more than one jammer tone is present, the jammer power is allocated uniformly over the set of tones. Thus, the power in each tone is equal to J/N, where N is the number of jammer tones.

The jamming components of the receiver I and Q samples are the result of passing the set of interference tones together with the signal and noise through the RF and IF circuits, downconverting to baseband, and sampling in integrate-and-dump circuits, whereupon the composite waveform is converted to digital values. The contribution of each jammer tone to these A/D samples can be written as

$$I_J = A\left(\frac{\sin \pi f_J \Delta t}{\pi f_J \Delta t}\right)\cos\phi_J$$
$$Q_J = A\left(\frac{\sin \pi f_J \Delta t}{\pi f_J \Delta t}\right)\sin\phi_J \tag{5.23}$$

where A is the amplitude of the jammer tone, f_J is the offset between the j-th tone frequency and the receiver's estimate of the signal carrier frequency, ϕ_J is the average phase shift of the j-th tone during the sampling time, and Δt is the receiver A/D sampling interval. The tone amplitude, frequency, and phase include the effects of jammer channel scintillation and jammer vehicle dynamics. The frequency offset and phase also include the effects of receiver frequency and phase tracking loop operation and carrier frequency hopping, when used.

The total jamming interference is obtained by summing the I and Q components computed from Equation 5.23 over all interference tones that are contained within the sampling bandwidth of the receiver. As indicated in Figure 4-1, it is assumed that lowpass filters precede the integrate-and-dump sampling circuits, and that these filters effectively remove frequency components that fall well outside the sampling band. Hence, with an A/D sampling rate of f_s, only those tones that lie within $\pm f_s$ of the current estimate of the carrier frequency are included in the summation. This places the filter cutoff at the first nulls of the *sinx/x* response of the sampling circuits.

5.6.3 Linear FM Jamming.

Another interference waveform that may be encountered is linear frequency modulation. This may be a continuous waveform with a sawtooth frequency sweep. More commonly, it is a pulsed waveform with a linear frequency sweep during the pulse duration. This latter waveform is generally referred to as a chirp pulse and is usually associated with precision radar.

A received chirp pulse can be represented mathematically as

$$p(t) = \mathrm{Re}\{A(t)\exp[j2\pi(f_c + 0.5Bt/T_p)t + j\theta(t)]\} \quad , \qquad -T_p/2 \le t \le T_p/2$$

where $A(t)$ is the received pulse amplitude including the effects of jammer channel fading, f_c is the center of the swept frequency band, B is the extent of the swept band, T_p is the pulse or sweep duration, and $\theta(t)$ is the phase including Doppler shift and phase scintillation. Upon differentiating the modulation phase with respect to time, it is seen that the frequency varies linearly from f_c - $B/2$ to f_c + $B/2$ over the sweep duration. Therefore, the time-bandwidth product or pulse compression ratio is BT_p, and the frequency sweep rate is B/T_p.

The frequency sweep rate can be positive or negative, producing an up-chirp or down-chirp waveform, respectively. This may be specified by giving a positive or negative value for the swept bandwidth B or for the pulse compression ratio.

After passing through the receiver RF and IF stages and downconversion to baseband, the jamming components of the I and Q samples at the outputs of the integrate-and-dump sampling circuits are obtained by integrating the chirp waveform over the sampling interval. In any sampling interval, six possible cases need to be considered, depending on the relative pulse timing. All six cases are treated in the following subsection on pulse jamming. Here we present the results for the case where a jammer pulse begins and ends during the A/D sampling period (case 5). The results for each of the other cases can be cast in terms of these results.

When the chirp pulse begins and ends during the receiver sampling period, the jamming components of the I and Q samples are found to be

$$I_J = \frac{A}{\Delta t}\sqrt{\frac{\pi}{2|a|}}\left\{\cos\Phi\left[C(\alpha+\beta)-C(\alpha-\beta)\right]+s\sin\Phi\left[S(\alpha+\beta)-S(\alpha-\beta)\right]\right\}$$

$$Q_J = \frac{A}{\Delta t}\sqrt{\frac{\pi}{2|a|}}\left\{s\cos\Phi\left[S(\alpha+\beta)-S(\alpha-\beta)\right]-\sin\Phi\left[C(\alpha+\beta)-C(\alpha-\beta)\right]\right\} \quad (5.24)$$

where A is the received pulse amplitude including the effect of jammer fading, Δt is the receiver A/D sampling interval, *C(x)* and *S(x)* are the Fresnel cosine and sine integrals (Abramowitz and Stegun, 1964), and

$$a = \frac{\pi B}{T_p} \quad , \quad \text{positive for up-chirp, negative for down-chirp}$$

$$s = \frac{a}{|a|} \quad , \quad +1 \text{ for up-chirp, -1 for down-chirp}$$

$$\Phi = \frac{\omega^2}{4a} - \phi$$

$$\alpha = \sqrt{\frac{\omega^2}{2\pi|a|}}$$

$$\beta = \sqrt{\frac{|a|T_p^2}{2\pi}}$$

The quantity ω is the difference in angular frequency between the jammer pulse and the receiver's estimate of the signal carrier frequency, and ϕ is the difference between the jammer phase and receiver local oscillator phase. Both of these quantities are evaluated at the center of the sampling integral and include the effects of jammer Doppler shift and receiver tracking loop operation as well as carrier frequency hopping if used.

5.6.4 Jamming In PN/PSK Receivers.

The foregoing equations apply directly to systems with and without frequency hopping. They apply in principle to direct-sequence spread-spectrum systems. However, in the case of tone jamming and chirp jamming, such application would require evaluation on a chip-by-chip basis. Typically, a very large number of PN code chips are present during each A/D sample, causing chip-by-chip computation of interference to be prohibitively time consuming.

To reduce the computational requirements in direct-sequence PN/PSK applications, an approximate statistical model is employed (Dana and Frasier, 1998). Fortunately, the large number of chips per sample that makes chip-by-chip computation unattractive also ensures the efficacy of a statistical approach. The basic assumption in this approach is that the PN code sequence is sufficiently long that there is essentially no correlation from one chip to the next.

It can be shown that the jamming components in the I and Q channels of a PN/PSK receiver can be represented by correlated random variables whose distribution approaches the Gaussian form when the number of chips per sample is large. It has been found that as few as 16 chips per sample provide statistics that are remarkably close to Gaussian. Since most applications involve hundreds to thousands of chips per sample, Gaussian statistics are ensured.

Generation of the I and Q components of the interference voltage resulting from a tone jammer is implemented by transforming a pair of uncorrelated zero-mean Gaussian variables into a correlated pair through a rotation of coordinates. The resulting algorithm can be written as follows:

$$\begin{aligned} I_J &= \cos\left(\phi + \frac{\omega\Delta t}{2}\right) g_1 - \sin\left(\phi + \frac{\omega\Delta t}{2}\right) g_2 \\ Q_J &= \sin\left(\phi + \frac{\omega\Delta t}{2}\right) g_1 + \cos\left(\phi + \frac{\omega\Delta t}{2}\right) g_2 \end{aligned} \qquad (5.25)$$

where ω is the angular frequency offset of the jamming tone, ϕ is the tone phase shift at the beginning of the sampling period Δt, and g_1 and g_2 are uncorrelated zero-mean Gaussian variables with unequal variances. The tone frequency ω and phase ϕ include the effects of jammer Doppler shift and receiver tracking loop operation as well as jammer phase scintillation.

The variances of the two Gaussian variables depend on the tone frequency offset and on the number of PN code chips per sample:

$$\sigma_1^2 = \frac{A^2}{2n_c}\left[\frac{\sin\left(\frac{\omega T_c}{2}\right)}{\frac{\omega T_c}{2}}\right]^2 \left(1 + \frac{\sin\omega\Delta t}{n_c \sin\omega T_c}\right)$$

$$\sigma_2^2 = \frac{A^2}{2n_c}\left[\frac{\sin\left(\frac{\omega T_c}{2}\right)}{\frac{\omega T_c}{2}}\right]^2 \left(1 - \frac{\sin\omega\Delta t}{n_c \sin\omega T_c}\right)$$

where A is the amplitude of the jammer tone including the effects of jammer channel fading, T_c is the PN code chip period, and n_c is the number of PN chips per A/D sample.

The total jamming interference is obtained by summing the I and Q voltages given by Equation 5.25 over all interference tones that fall within the system bandwidth, which extends across the entire bandwidth of the PN code.

These equations are also applied to the case of chirp pulse jamming in PN/PSK receivers by computing an effective center frequency and phase for the portion of the chirp pulse that is present during the A/D sampling interval.

5.6.5 Pulse Jamming.

Any of the jamming waveforms can be continuous or pulsed. A pulse jammer is specified by the pulse repetition frequency (PRF) and the duty cycle. The duty cycle is the ratio of pulse duration T_p to pulse repetition interval (reciprocal of the PRF).

The jamming components of the I and Q samples are obtained by integrating the jamming waveforms over the receiver sampling interval Δt. For a pulse jammer, this integration extends only over the period of time that the jammer pulse is actually present during a given sampling interval.

Define k to be a receiver sampling index. Then if the receiver clock ran at a constant rate, the sampling times would be $k\Delta t$. However, because of receiver time tracking, the receiver clock may be advanced or retarded as it attempts to maintain synchronization with the signal timing. Changes in the receiver delay estimate affect the relative timing between receiver samples and jammer pulses. This relative timing is important because a pulse jammer may be able to walk the receiver timing off of the signal, causing loss of symbol synchronization.

In any given sampling interval, six possible timing cases must be considered:

1) No jamming pulse is present at any time during the sampling interval;
2) A jamming pulse is continually present throughout the sampling interval;
3) A jamming pulse ends during the sampling interval;
4) A jamming pulse begins during the sampling interval;
5) A jamming pulse begins and ends during the sampling interval;
6) A jamming pulse ends and the next one begins during the sampling interval.

We rule out any other possibilities by not allowing the jammer PRF to exceed the receiver A/D sampling rate. It is seen that case 6 is a combination of cases 3 and 4 within the same sampling interval.

For the case of pulsed noise, these six cases are readily handled by adjusting the jammer noise variance for each sample based on the amount of time that the pulse is present during the sampling interval.

Similarly for pulsed tones, the six cases simply require adjustments to the amplitude A and integration time Δt in Equation 5.23 for each sample depending on the time that the pulse is present.

For a chirp pulse, Equation 5.24 is directly applicable to case 5. This equation can also be applied to the other timing cases with suitable modifications to the integration time, the center of the swept frequency band, and the phase shift. Specifically, the pulse duration T_p is replaced with the time interval that the pulse is present during the sample, and the offset frequency ω and phase shift ϕ are evaluated at the center of this time interval.

5.6.6 Frequency Follower Jammer.

Pulse jammers can be frequency followers, if the communication system uses frequency hopping. Between pulses when the jammer is not radiating, a frequency follower listens to the transmitted communications signal. It then radiates the next jamming pulse at or near the observed system frequency.

To be effective, the jammer must be close enough to the signal path between the communications transmitter and receiver that the jammer pulse arrives at the receiver while the observed signal is still being received. If the jammer is too far away, the system will have hopped to a new frequency before the jammer pulse arrives, rendering it completely ineffective.

The time delay between the signal and the jammer is determined from a range offset that is part of the jammer dynamics capability.

5.6.7 Jammer Dynamics.

Jammer Doppler and delay dynamics can result from platform motion. Three canonical range dynamics profiles are available: sinusoidal, initial-value, and square-wave profiles. For each profile the range offset, velocity, and acceleration can be specified, along with a period for the sinusoidal or square-wave profiles or a duration for the initial-value profile.

The specified velocity and acceleration vary with time in accordance with the selected profile and are integrated to yield a range offset and velocity time history. The resulting values of range and velocity determine the jammer delay and Doppler shift as a function of time.

When the jammer has frequency follower capability, the jammer range offset takes on special significance as noted above. The range offset determines the time delay between the communications signal path and the jammer path. If this delay exceeds the frequency-hop period, frequency following will not be effective.

5.6.8 Jammer Antenna Rotation.

The jammer antenna can be fixed in position relative to the communications receiver, or it can be rotating. A rotating antenna can be used to simulate the effects of periodic interference such as that produced by a friendly or hostile radar.

When a rotating jammer antenna is specified, the interference as a function of time depends on the rotation speed, antenna beamwidth and gain pattern. The effective beamwidth is the angular separation between the half-power points on the jammer radiation pattern as measured at the communications receiver. A *sin(x)/x* gain pattern is normally used to characterize the jammer antenna pattern. This can be changed to a user-specified gain pattern if desired.

5.6.9 Jammer Scintillation.

The propagation path between the jammer and the communications receiver can be fading or nonfading, independent of the signal propagation path. The intensity of jammer fading is determined by the S_4 scintillation index. The scintillation decorrelation time determines the fading rate. Refer to the discussion of the channel model for more information concerning these parameters.

The jammer model is implemented in subroutines JAMMR0, JAMMR1, JAMMR2, and JAMMR3. The module selection depends on which communications receiver is involved. Each of these jammer modules is initialized by subroutine IJAMMR. Each module is supported by subroutine JAMDYN, which handles details of pulse timing and jammer dynamics.

SECTION 6

REFERENCES

Abramowitz, M., and I. A. Stegun, *Handbook of Mathematical Functions*, National Bureau of Standards, June 1964 (UNCLASSIFIED).

Bogusch, R. L., F. W. Guigliano, D. L. Knepp, and A. H. Michelet, "Frequency Selective Propagation Effects on Spread-Spectrum Receiver Tracking," *Proceedings of the IEEE*, Vol. 69, No. 7, pp. 787-796, July 1981 (UNCLASSIFIED).

Bogusch, R. L., F. W. Guigliano, and D. L. Knepp, "Frequency-Selective Scintillation Effects and Decision Feedback Equalization in High Data-Rate Satellite Links," *Proceedings of the IEEE*, Vol. 71, No. 6, pp. 754-767, June 1983 (UNCLASSIFIED).

Bogusch, R. L., *An Approach to the Development of Software for Computer-Aided Design of Satellite Communications Receivers for Operation in Scintillating Channels*, DNA-TR-89-109, MRC-R-1231, Mission Research Corporation, April 1989a (UNCLASSIFIED).

Bogusch, R. L., *Digital Communications in Fading Channels: Modulation and Coding*, AFWL-TR-87-52, MRC-R-1043, Mission Research Corporation, April 1989b (UNCLASSIFIED).

Bogusch, R. L., *Digital Communications in Fading Channels: Tracking and Synchronization*, WL-TR-90-15, MRC-R-1251, Mission Research Corporation, April 1990 (UNCLASSIFIED).

Bogusch, R. L., and A. H. Michelet, *Digital Communications in Fading Channels: Signal Acquisition*, PL-TR-92-1024, MRC-R-1365, Mission Research Corporation, March 1993 (UNCLASSIFIED).

Bogusch, R. L., *Digital Communications in Fading Channels: Computer-Aided Design and Evaluation*, DNA-TR-95-120, MRC-R-1504, Mission Research Corporation, May 1996 (UNCLASSIFIED).

Cahn, C. R., *et al.*. "Software Implementation of a PN Spread Spectrum Receiver to Accommodate Dynamics," *IEEE Trans. Commun.*, Vol. COM-25, pp. 832-840, August 1977 (UNCLASSIFIED).

Cross, E., *NELS II Operation & Evolution*, DNA-TR-87-206, General Electric Company, July 1987 (UNCLASSIFIED).

Dana, R. A., *Temporal Statistics of Scintillation for Satellite Communication and Radar Systems*, DNA-TR-81-129, MRC-R-692, Mission Research Corporation, April 1982 (UNCLASSIFIED).

Dana, R. A., *Propagation of RF Signals Through Structured Ionization, the General Model*, DNA-TR-90-9, MRC-R-1262R, Mission Research Corporation, March 1991 (UNCLASSIFIED).

Dana, R. A., *Statistics of Sampled Rician Fading*, DNA-TR-92-98, MRC-R-1410, Mission Research Corporation, February 1993 (UNCLASSIFIED).

Dana, R. A., *A Non-Stationary Model for Frequency Selective Channels*, MRC-R-1403R, Mission Research Corporation, September 1994 (UNCLASSIFIED).

Dana, R. A., *Advanced Channel Simulator Qualification Test Plan*, MRC-R-1491, Mission Research Corporation, March 1995 (UNCLASSIFIED).

Dana, R. A., B. R. Milner, and R. L. Bogusch, "Effects of Channel Tap Spacing on Delay Lock Tracking," *SPIE Conference on Wireless Data Transmission*, October 1995 (UNCLASSIFIED).

Dana, R. A., and R. L. Bogusch, *Digital Communications in Fading Channels: COMLNK Validation*, DSWA-TR-98-61, MRC-R-1531, July 1998 (UNCLASSIFIED).

Dana, R. A., and S. M. Frasier, Mission Research Corporation, Private Communication, June 1998 (UNCLASSIFIED).

Dodson, R. E., and F. W. Guigliano, *PRPSIM Users Manual - Version 3*, MRC-R-1538 (to be published by DTRA), July 1998 (UNCLASSIFIED).

Heller, J. A., and I. M. Jacobs, "Viterbi Decoding for Satellite and Space Communication," *IEEE Trans. Commun.*, Vol. COM-19, pp. 835-848, October 1971 (UNCLASSIFIED).

Jacobs, I. M., "Practical Applications of Coding," *IEEE Trans. Inform. Theory*, Vol. IT-20, pp. 305-310, May 1974 (UNCLASSIFIED).

Knepp, D. L., "Multiple Phase-Screen Calculation of the Temporal Behavior of Stochastic Waves," *Proceedings of the IEEE*, Vol. 71, No. 6, pp. 722-737, June 1983 (UNCLASSIFIED).

Larsen, K. J., "Short Convolutional Codes With Maximal Free Distance for Rates 1/2, 1/3, and 1/4," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 371-372, May 1973 (UNCLASSIFIED).

Lee, P. J., "New Short Constraint Length, Rate 1/N Convolutional Codes Which Minimize the Required SNR for Given Desired Bit Error Rates," *IEEE Trans. Commun.*, Vol. COM-33, pp. 171-177, February 1985 (UNCLASSIFIED).

Lee, P. J., "Further Results on Rate 1/N Convolutional Code Constructions with Minimum Required SNR Criterion," *IEEE Trans. Commun.*, Vol. COM-34, pp. 395-399, April 1986 (UNCLASSIFIED).

Lindsey, W. C., *Synchronization Systems in Communication and Control*, Prentice-Hall, 1972 (UNCLASSIFIED).

Newman, D. D., *Observations of the Nature and Effects of Interleavers in Communications Links*, AFWL-TR-82-125, MRC-R-704, Mission Research Corporation, February 1983 (UNCLASSIFIED).

Odenwalder, J. P., *Dual-k Convolutional Codes for Noncoherently Demodulated Channels*, International Telemetering Conf., Vol. XII, 1976 (UNCLASSIFIED).

Oppenheim, A. V., and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, 1975 (UNCLASSIFIED).

Papoulis, A., Probability, *Random Variables, and Stochastic Processes*, McGraw-Hill, 1965 (UNCLASSIFIED).

Proakis, J. G., *Digital Communications*, McGraw-Hill, 1983 (UNCLASSIFIED).

Viterbi, A. J., *Principles of Coherent Communication*, McGraw-Hill, 1966 (UNCLASSIFIED).

Viterbi, A. J., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inform. Theory*, Vol. IT-13, pp. 260-269, April 1967 (UNCLASSIFIED).

Viterbi, A. J., and J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, 1979 (UNCLASSIFIED).

Wittwer, L. A., *Radio Wave Propagation in Structured Ionization for Satellite Applications*, DNA 5304D, Defense Nuclear Agency, December 1979 (UNCLASSIFIED).

Wittwer, L. A., *A Trans-Ionospheric Signal Specification for Satellite C3 Applications*, DNA 5662D, Defense Nuclear Agency, December 1980 (UNCLASSIFIED).

Wittwer, L. A., *Radio Wave Propagation in Structured Ionization for Satellite Applications II*, DNA-IR-82-02, Defense Nuclear Agency, August 1982 (UNCLASSIFIED).

Wittwer, L. A., *Radio Wave Propagation in Structured Ionization for Satellite and Radar Applications*, DNA-IR-93-35, Defense Nuclear Agency, August 1993 (UNCLASSIFIED).

APPENDIX

INSTALLATION AND QUICK START MANUAL

A.1 SYSTEM REQUIREMENTS.

This set of diskettes contains executable and data files for the current release of COMLNK, Version 5.3. System requirements for this version are essentially unchanged from previous releases, except that the user interface now includes mouse support. As before, COMLNK requires a PC with the following installed hardware and software:

Processor:	80386 or higher (486DX, Pentium, 32-bit Intel-compatible)
Coprocessor:	80387 or higher (486DX, Pentium, 32-bit Intel-compatible)
Memory:	8 MB or more available RAM (extended memory)
Hard Disk:	8 MB or more available disk storage
Floppy Disk:	3.5-inch high density (1.44 MB)
Display:	VGA or better, color monitor required
Keyboard:	84-key or better
Mouse:	Supported but not required
Operating System:	DOS 3.3+, Windows 3.1, Windows 95/98, Windows 2000 (not compatible with NT 4.x)

COMLNK is intended to be run from the DOS prompt. It can be run under Windows 3.1 enhanced mode and under Windows 95, Windows 98, and Windows 2000.[1] It may be necessary to use the Windows PIF editor and allocate extended memory to run under Windows 3.1. We have not encountered any problems with Windows 95/98.

When running under Windows 95, we recommend that the 8x12 font be selected for the DOS window or that a full screen display be used. This provides proper display of mathematical symbols when viewing files in the Info menu. True Type fonts may not properly display math symbols in the DOS window in some Windows 95 installations. No difficulties have been reported with Windows 98.

A.2 INSTALLATION.

Installation of COMLNK is simply a matter of copying all of the files on the distribution diskettes to the hard disk. It is a good idea to first create a new directory on the hard disk, change to that directory, and then copy the files from the diskettes. For example, the following sequence of commands can be entered at the DOS prompt:

[1] COMLNK is not compatible with previous versions of the Windows NT operating system. A conflict with the DOS extender, supplied by the compiler manufacturer, prevents COMLNK from loading under Windows NT 4.x. Use of Windows 95/98 or upgrading to Windows 2000 is recommended.

```
C:
CD\
MD COMLNK
CD COMLNK
(insert disk 1 into drive A)
COPY A:*.*
(remove disk 1, insert disk 2)
COPY A:*.*
(remove disk 2, insert disk 3)
COPY A:*.*
```

Change A: to B: in these commands if the 3.5-inch floppy drive is drive B. These commands will create a directory named COMLNK in the root directory of the hard drive (assumed to be drive C). Then the files from each diskette will be copied into the new COMLNK directory. A different name can be chosen for the directory if desired. Just be sure to copy the files from each of the distribution diskettes to the same directory on the hard drive.

When upgrading from a previous release, either copy this new release into the same directory, overwriting older files, or create a new directory with a different name for this release.

If the DTRA (formerly DSWA) ASSIST program has been installed, the version of COMLNK supplied with ASSIST can be updated by copying this new release into the existing COMLNK subdirectory under the ASSIST directory. The current release of COMLNK will then be available when ASSIST is run.

In any case, the following files will be copied to the hard disk by the installation procedure outlined above.

Disk 1:

COMLNK.EXE	program executable file
DEMO.DAT	data file for set of demonstration cases
ACQUIRE.DAT	data file for example of FSK frame acquisition
BENTPIPE.DAT	data file for example of bent-pipe transponder
CHAINED.DAT	data file for example of linked cases
COMLNK.DAT	data file for default example
COQPSK.DAT	data file for example of coherent offset QPSK link
CPSK.DAT	data file for example of coherent PSK link
DPSK.DAT	data file for example of differentially coherent PSK link
DQPSK.DAT	data file for example of differentially coherent QPSK link
8FSK.DAT	data file for example of 8-ary FSK link
EXEQUIT.DAT	data file illustrating use of .EXE and .QUIT directives
FHCPSK.DAT	data file for example of FH/CPSK link
FHDPSK.DAT	data file for example of FH/DPSK link
FHFSK.DAT	data file for example of FH/FSK frequency acquisition
FHFSK_IT.DAT	data file for example of independent-tone FH/FSK link
JAM16FSK.DAT	data file for example of jammer with FH/FSK
JAMPNPSK.DAT	data file for example of jammer with PN/PSK
L1_CA.DAT	data file for example of single-channel L1 C/A receiver
PNCPSK.DAT	data file for example of PN/CPSK link
PNDPSK.DAT	data file for example of PN/DPSK link
RECORDED.DAT	data file for example of link with recorded channel
RECORDED.CIQ	example channel data file for RECORDED.DAT

SCENARIO.DAT	data file for example of link with time-varying channel
SCENARIO.CHN	example channel parameter file for SCENARIO.DAT
SCRIPT.DAT	data file for example of scripted cases
SCRIPT.RUN	example script file for SCRIPT.DAT
TWOLINKS.DAT	data file for example of two independent links
UPDOWN.DAT	data file for example of concatenated uplink/downlink
UPXDOWN.DAT	data file for example of uplink/crosslink/downlink
2JAMMERS.DAT	data file for example of link with two jammers
2RELAYS.DAT	data file for example of et1-> sat1-> et2-> sat2-> et3
2XPONDR.DAT	data file for example of double transponder relay link
SCALE_.RUN	example script file to optimize quantizer scale factor
VS_EBN0.RUN	example script file for performance vs. Eb/No
VS_EBN01.RUN	example script file for performance vs. Eb/No
VS_EBN02.RUN	example script file for performance vs. Eb/No
VS_FZERO.RUN	example script file for performance vs. f0
VS_TAU0.RUN	example script file for performance vs. tau0
ACQUIRE.XMT	transmit message file for ACQUIRE.DAT
BENTPIPE.XMT	transmit message file for BENTPIPE.DAT
COMLNK.XMT	transmit message file for COMLNK.DAT
COQPSK.XMT	transmit message file for COQPSK.DAT
CPSK.XMT	transmit message file for CPSK.DAT
DEMO00.XMT	transmit message file for DEMO.DAT
DEMO01.XMT	"
DEMO02.XMT	"
DEMO03.XMT	"
DEMO04.XMT	"
DEMO05.XMT	"
DEMO06.XMT	"
DEMO07.XMT	"
DPSK.XMT	transmit message file for DPSK.DAT
DQPSK.XMT	transmit message file for DQPSK.DAT
8FSK.XMT	transmit message file for 8FSK.DAT
FHCPSK.XMT	transmit message file for FHCPSK.DAT
FHDPSK.XMT	transmit message file for FHDPSK.DAT
FHFSK.XMT	transmit message file for FHFSK.DAT
FHFSK_IT.XMT	transmit message file for FHFSK_IT.DAT
JAM16FSK.XMT	transmit message file for JAM16FSK.DAT
JAMPNPSK.XMT	transmit message file for JAMPNPSK.DAT
L1_CA.XMT	transmit message file for L1_CA.DAT
PNCPSK.XMT	transmit message file for PNCPSK.DAT
PNDPSK.XMT	transmit message file for PNDPSK.DAT
RECORDED.XMT	transmit message file for RECORDED.DAT
SCENARIO.XMT	transmit message file for SCENARIO.DAT
SCRIPT.XMT	transmit message file for SCRIPT.DAT
2JAMMERS.XMT	transmit message file for 2JAMMERS.DAT
2RELAYS.XMT	transmit message file for 2RELAYS.DAT
2XPONDR.XMT	transmit message file for 2XPONDR.DAT
ACQUIRE.PRN	print output file from running ACQUIRE.DAT
BENTPIPE.PRN	print output file from running BENTPIPE.DAT
COMLNK.PRN	print output file from running COMLNK.DAT
COQPSK.PRN	print output file from running COQPSK.DAT
CPSK.PRN	print output file from running CPSK.DAT
DEMO.PRN	print output file from running DEMO.DAT
DEMO2.PRN	"
DEMO3.PRN	"
DEMO4.PRN	"

DEMO5.PRN	"
DEMO6.PRN	"
DEMO7.PRN	"
DPSK.PRN	print output file from running DPSK.DAT
DQPSK.PRN	print output file from running DQPSK.DAT
8FSK.PRN	print output file from running 8FSK.DAT
FHCPSK.PRN	print output file from running FHCPSK.DAT
FHDPSK.PRN	print output file from running FHDPSK.DAT
FHFSK.PRN	print output file from running FHFSK.DAT
FHFSK_IT.PRN	print output file from running FHFSK_IT.DAT
JAM16FSK.PRN	print output file from running JAM16FSK.DAT
JAMPNPSK.PRN	print output file from running JAMPNPSK.DAT
L1_CA.PRN	print output file from running L1_CA.DAT
PNCPSK.PRN	print output file from running PNCPSK.DAT
PNDPSK.PRN	print output file from running PNDPSK.DAT
RECORDED.PRN	print output file from running RECORDED.DAT
SCENARIO.PRN	print output file from running SCENARIO.DAT
SCRIPT.PRN	print output file from running SCRIPT.DAT
SCRIPT2.PRN	"
SCRIPT3.PRN	"
SCRIPT4.PRN	"
SCRIPT5.PRN	"
TWOLINKS.PRN	print output file from running TWOLINKS.DAT
UPDOWN.PRN	print output file from running UPDOWN.DAT
UPXDOWN.PRN	print output file from running UPXDOWN.DAT
2JAMMERS.PRN	print output file from running 2JAMMERS.DAT
2RELAYS.PRN	print output file from running 2RELAYS.DAT
2XPONDR.PRN	print output file from running 2XPONDR.DAT
COMLNK#I.INF	system file for Info menu, Introduction

Disk 2:

COMLNK#H.INF	system file for Info menu, How do I ...
COMLNK#P.INF	system file for Info menu, Channel model
COMLNK#F.INF	system file for Info menu, Fortran source

Disk 3:

COMLNK#U.INF	system file for Info menu, User interface
COMLNK#M.INF	system file for Info menu, Digital comm.
COMLNK#A.INF	system file for Info menu, Assembly source
BIN2PLT.EXE	plot file conversion utility program
COMLNK.BAT	example DOS batch file (see discussion below)
CHANGE.TXT	list of changes in each release of COMLNK
README.TXT	this file

COMLNK can be run from the directory in which these files are placed. It can also be run from a different directory if the COMLNK directory is added to the DOS path. Another way to run COMLNK from any directory is to place a small DOS batch file named COMLNK.BAT in any directory that is already on the DOS path. Type PATH at the DOS prompt to determine which directories are on the path, and then store COMLNK.BAT in any one of these directories. The COMLNK.BAT file should contain lines similar to the following:

```
@ECHO OFF
C:\COMLNK\COMLNK %1 %2
```

An example COMLNK.BAT file is included on the distribution diskettes. This example assumes that the hard disk is drive C and that the files were copied from the distribution diskettes to a directory named COMLNK as suggested above. If a different directory name or path was chosen, change the first occurrence of COMLNK to the name of the directory (including the path) in which the files were copied.

A.3 GETTING STARTED.

Once the files have been copied from the distribution diskettes to the hard disk, any of the examples can be run by entering the following command at the DOS prompt:

```
COMLNK filename
```

The filename following COMLNK is optional. If present, it should be one of the data files (include the .DAT extension in filename). The filename is not case sensitive; use either upper or lower case. Once the Enter key is pressed, the program will load and display an opening title screen. This screen will remain for about 7 seconds and then disappear. The title will disappear sooner if any key is pressed while it is on display.

While the title screen is displayed, the program reads the data file with the name that was entered on the command line. If no filename was entered, the file COMLNK.DAT is read in the first time the program is started. In any case, the first link layout in the file will be displayed when the title screen goes away. To run this layout, open the Run menu and select execute layout.

If a blank screen with the message "File not found" appears, either the filename was mistyped or the data file is in a different directory. In either case open the File menu and select Open layout file. If the current directory contains the files from the distribution diskettes, a list of data files will appear. If not, the file specification at the bottom of the screen can be edited to access the directory that contains the files. Directories can be changed using a file specification `*.*` which shows all files, including directories. Highlight the desired directory and press Enter to switch to that directory.

Once a list of data files is displayed, move the highlight to the name of the desired file and press Enter. The file will be read in and can be run by opening the Run menu and selecting execute layout.

We suggest that the set of demonstration cases be run first to get an idea of what the program does, especially if you are a first-time user. To run the demonstration, select the file DEMO.DAT from the File menu or simply enter the following command at the DOS prompt:

```
COMLNK DEMO.DAT
```

If the current directory is the one that contains the data files from the distribution diskettes, the first link layout in the demonstration file will appear when the title screen goes away. Open the Run menu and select execute layout to run the demo.

A.4 RUNNING TIME.

The total run times for the set of demonstration cases and for the default COMLNK.DAT file are tabulated below for four PCs. These include a 400-MHz Pentium II machine, a 233-MHz Pentium, a 150-MHz Pentium, and a 100-MHz 486DX4 notebook PC. All of the print files included on the distribution diskettes were produced with the 400-MHz Pentium II, and the resulting CPU times are provided at the end of each file.

Run Times on Several PCs				
Data File	**Pentium-400 512K Cache**	**Pentium-233 256K Cache**	**Pentium-150 256K Cache**	**486DX4-100 No Cache**
DEMO.DAT	1.1 min	2.1 min	3.8 min	11.7 min
COMLNK.DAT	3.7 sec	6.3 sec	10.5 sec	33.8 sec

Within a given chip family, run times scale approximately inversely with clock speed, although the cache size and floating-point unit (FPU) design also have an effect. Compared to the 486, the Pentium provides a speed advantage because of its dual-pipeline processor and improved FPU. After scaling for clock speed, run times on a Pentium average around a factor of 1.5 to 2.0 less than on a 486. The Pentium II provides an additional speed advantage because of its larger cache and faster FPU.

COMLNK has been found by several users to be a good test of whether their computer is functioning properly. If run times are significantly longer than the foregoing data would suggest, you may wish to investigate your hardware configuration settings. Among other things, check that the CPU speed is set correctly. Most PCs enable the clock speed to be changed either manually or via setup software. On some machines, this may require checking that jumpers on the motherboard are positioned properly. Also, check the RAM speed and whether the secondary (L2) RAM cache is enabled.

Two users have reported that their machines were set up incorrectly by their suppliers. Consequently, COMLNK has been employed as a benchmark to help evaluate competing machines during procurement. In this context it should be noted that COMLNK is CPU-bound and not I/O-bound.

A.5 NAVIGATING THE MENUS.

The user can move around in this program in several ways. The traditional interface uses only the keyboard. Use the arrow keys to move the menu highlight to the desired menu, and then press the Enter key to open the menu. Once a menu is open, use the arrow keys to select a function, and press Enter to execute that function. Menus can be changed while they are open to examine the functions available in each. Press the Esc key to cancel a function or close a menu; Esc backs out of any operation at any time.

Alternatively, "hot keys" can be used to open menus and execute functions. Active hot keys are always displayed in red in the menus. When no menu is open, use the hot keys on the top menu bar to open a menu. Once a menu is open, use the hot keys in the menu list to execute one of the functions. The effect is the same as moving the highlight and pressing Enter. Again, Esc will cancel any operation or close a menu.

Control key combinations further simplify moving around. By simultaneously pressing one of the Ctrl keys together with a letter key, the function associated with that letter can be executed without first opening the menu containing the function. Control key assignments are the same as the hot keys displayed in red in the open menu lists (except not the Info menu). In most cases, the key is the first letter in the function name. For example, to Open a file from anywhere in the menu structure, press Ctrl-O. To Browse any file, press Ctrl-B. To Save the current layout, press Ctrl-S. To View a large layout, press Ctrl-V. To Quit the program, press Ctrl-Q. Many other control key combinations are found by examining the open menu lists.

At any time, press the F1 key for context-sensitive help. We strongly recommend frequent use of this feature. The help messages are intended to familiarize the user with the methods of moving around the menu system. Help also provides brief descriptions of each menu function. When examining module data in the Layout menu, F1 provides a brief explanation of each parameter. In all cases, F1 toggles the on-screen help on and off. Once on, help is displayed until it is cancelled with F1. Esc also cancels help, as does executing a run.

Menus can be opened and help messages can be displayed while the program is running, without slowing it down. In fact, many things can be done with the program while it is executing a layout without significantly affecting its speed. We suggest that the user experiment by opening menus, reading the help messages, and trying out each function.

A mouse or other pointing device can also be used to navigate the menus and control program operation. Following standard convention, the left mouse button provides primary control. The right mouse button toggles on-screen help on or off in the same manner as the F1 key.

Any combination of mouse and keyboard control can be used at any time. When a mouse is available, scroll arrows appear at the right side and bottom of the layout screen display. Clicking on the scroll arrows moves the layout around on the screen in the same manner as the Display menu View function.

Any of the menus can be opened by clicking on the menu bar at the top of the screen. Click on any item in an open menu list to execute that function. When a list of file names appears in a menu list, click on the name of a file to highlight it, or double-click on it to open the file. When keyboard commands are shown in the status bar at the bottom on the screen, click on any command to execute it in the same manner as if that key had been pressed.

When a menu is open or an operation is awaiting a user response, clicking on the layout area outside of the menu area closes the menu or backs out of the operation in the same manner as pressing the Esc key. The exception to this rule is when an open submenu contains the Cancel option; in that case, the user must click on Cancel (or one of the other submenu options) to get out of that submenu.

Any time that a layout is visible on the screen, the user can click on any module in the layout to open its data menu. It may be necessary to click on the module more than once if another function is open, so as to back out of that function and open the module data menu. Once a data menu is open, use scroll arrows in the data menu or click on a menu item to move the highlight toward that item. (The bottom scroll arrow may be obscured by a submenu symbol, but it is still active.) When a desired data item is highlighted, click on it to open the data entry field. After typing in the data, either press the Enter key or click on the Enter command at the bottom of the screen to enter the new data. Clicking anywhere else cancels the data entry.

A.6 EDITING LAYOUTS.

The layout from any file can be edited, or a completely new layout can be built using any combination of available modules. Files can be opened and read in and layouts can be edited and saved while a run is in progress.

Use the Layout menu to edit a layout. Examine parameter values with the Module data function. Move the module highlight to each module and press Enter to open a data menu. Move the data highlight to each quantity and press F1 for a brief description. The data in any layout can be examined even when it is running. The data can be changed only if the layout is not running. To change data in a running layout, select Edit from the File menu to make a copy and then select Module data to edit the copy. With a data menu open, press Enter to change a value. A data entry field in which a new value can be typed will appear at the bottom of the screen. Press Enter when finished typing to replace the previous value with the new value, if it is within an acceptable range. Press Esc to cancel the data entry and leave the value unchanged.

Select Insert module in the Layout menu for a list of available modules. Build or edit a layout using the Insert, Delete, Replace, Module data, and Copy/cut/paste functions. In most cases, a given type of module can be used as many times as desired in a layout. For example, it is a simple matter to design a link with concatenated coding. Each instance of the module can be given the same or different design data. Concatenated links and multiple links are easily built. The example data files illustrate the types of links that can be constructed. This program has been extensively applied in the design and evaluation of rather complex communications networks.

When finished editing a layout, go to the Run menu and select the Analyze layout function. This will check the design for inconsistencies or inappropriate module placement. If a problem is detected, a warning is displayed and the user is given the option of correcting the problem or ignoring the warning. While this analysis capability will catch many errors in a layout, it is not foolproof. In all cases, the user should carefully examine the module data and performance results provided in the print output file.

Please note that passing the analysis is not a guarantee of good link performance. It will probably be necessary to experiment with a new design to find one that performs well over the range of channel characteristics that pertain to the application. This program does not attempt to automatically find such a design. Rather, it attempts to make the user's efforts in this regard more efficient.

A.7 SAVING LAYOUTS.

To save a layout, go to the File menu and select Save this layout. A filename will be suggested. To change the name, select Rename and edit the filename to be whatever is desired. Press F1 for help in the editing process. Data files may have any extension, not just .DAT. Different extensions can be used to organize data among several projects, for example.

Do not worry about overwriting a file; the program will automatically back up an old file having the same name by renaming it with a slightly modified extension. Even backups are saved in this way. No data is lost by saving a layout, no matter how many times the same name may be used. If a layout has been changed but not saved, the program will save it automatically before quitting using the name AUTOSAVE.DAT.

Using the menus is by far the easiest way to edit and save layout data files. However, layout data files can also be edited off-line using any text editor that can read and write plain ASCII files. The COMLNK.DAT and DEMO.DAT files contain information for those who wish to prepare layout files off-line.

A.8 SCRIPTING RUNS.

There are two ways to set up a series of runs. One way is to set up a main data file that contains data for the first case in a series. This file then transfers control to a script file that contains just the data that changes for subsequent cases. The files SCRIPT.DAT and SCRIPT.RUN illustrate this method. The main file can be created and edited using the Layout menu. At the present time, however, the script file must be created and edited using an off-line ASCII text editor. Several example script files are included on the distribution diskettes. We plan to improve the scripting process in a future release.

There is a second way to create a series of runs, using the program as it currently stands. Use the Layout menu to create or edit the file for the first case in a series. While in the Layout menu select Module data, open the data menu for any modem in the layout, then open the run options submenu at the bottom of the modem data menu. Enter the name of a file that will contain the data for the next case and toggle the script file status to active. When finished entering data for the first case, save the file under some name. Then simply make any changes necessary for the second case, including the name of a file that will contain the next case, and save it under the name to which the preceding case refers. In this way, an unlimited number of cases can be linked together, each in its own file.

With either method of scripting a series of runs, the first file in the series must be saved to disk prior to running the script. The script is activated by reading this main file in from disk and then executing it.

A.9 WHAT'S NEW.

Version 5.3 provides a rather diverse set of enhancements. Principal changes include mouse support, a new FSK acquisition module, new multiplexing options, and a test for catastrophic codes. Each of these enhancements is described in more detail in the following paragraphs.

Mouse support has been incorporated. COMLNK now recognizes when the host machine has a mouse or other pointing device and responds to that device. Mouse support is available in either pure DOS mode or when running in a DOS box under Windows. In either environment the mouse is now available as an alternative to the keyboard when opening menus and selecting menu options. The keyboard interface has not been changed and can still be used with or without a mouse. When using a mouse, a layout can be scrolled on the screen by clicking on control arrows at the right and bottom of the display. Data menus can be opened simply by clicking on a module in a layout. Files can be selected by clicking on the filename in a directory listing. Key commands listed in the bottom status line on the screen can be executed by pointing to the command and clicking on it. On-line help can be turned on or off at any time by right-clicking the mouse.

A new acquisition module for unhopped M-ary FSK links has been incorporated. There are now two FSK acquisition modules. The new module pair, ACQBTX and ACQBRX, provides an alternative design approach to acquisition of preamble frames in unhopped M-ary FSK links. The new acquisition module uses hard or soft binary correlators to detect the preamble sequence. The preamble structure and design values are provided through the user interface.

New interspersing options have been incorporated for multiplexing sync symbols and for general time-division multiplexing applications. In addition to the options previously available (uniform interspersing or pseudorandom permutation of transmission frames), sync multiplexing and general TDM can now employ two other types of nonuniform interspersing. The symbols to be multiplexed (sync or other user data) can be grouped together at either the front or the rear of each frame. The new multiplex/demultiplex modules can be used to stuff extra bits or symbols into any type of link at any point.

A test for catastrophic codes has been incorporated in the convolutional code data menu. A catastrophic code is one for which the Viterbi decoder may lock in an erroneous state and fail, depending on signal-to-noise ratio and channel conditions. With such a code, the decoder can produce an unbounded number of data errors with a finite number of demodulation errors. The probability of catastrophic failure increases with decreasing SNR; it is especially likely to occur in a fading channel. A necessary and sufficient condition for a code to be catastrophic is that the generator polynomials all have a common factor. This test is now implemented in the COMLNK code data menu and a warning is issued if such a code is specified. Another warning is displayed if the user tries to run the code. The warnings can be ignored and the code can be run, if desired. For an example, open the COMLNK.DAT file and change the code generators to 135, 145, 171.

The foregoing paragraphs summarize changes incorporated in the current release. For a more complete list of changes dating back to the first beta release, see the accompanying change history file CHANGE.TXT.

A.10 WHAT'S MISSING.

While a number of enhancements have been introduced as noted above, there are several other waveforms, codes, and signal-processing functions that we plan to incorporate as time permits.

In addition, the user interface still lacks the capability to display results of modem operation and channel characteristics graphically. A plot output file can be turned on in the modem data menu, and an extensive plot data selection menu is available, but the associated on-line graphics routines are not yet installed. However, a file conversion utility program, BIN2PLT.EXE, is provided to allow the user to use his or her own plot package. See the help topic on how to plot, which is accessed from the "How do I ..." section in the Info menu.

A.11 DOCUMENTATION.

An on-line user's manual is available by opening the Info menu. All or part of this manual, which is the basis for this document, can be printed using a word processor. See the help topic on how to print user manual, which is accessed from the "How do I ..." section in the Info menu.

Formal documentation of COMLNK requires several volumes. We suggest that all users obtain copies of the following technical reports. The first four reports provide detailed descriptions of the simulation techniques employed in the program. The fifth report documents some of the results of the verification and validation process.

1. Bogusch, R. L., *Digital Communications in Fading Channels: Modulation and Coding*, AFWL-TR-87-52, MRC-R-1043, April 1989.
2. Bogusch, R. L., *Digital Communications in Fading Channels: Tracking and Synchronization*, WL-TR-90-15, MRC-R-1251, April 1990.
3. Bogusch, R. L., and A. H. Michelet, *Digital Communications in Fading Channels: Signal Acquisition*, PL-TR-92-1024, MRC-R-1365, March 1993.
4. Bogusch, R. L., *Digital Communications in Fading Channels: Computer-Aided Design and Evaluation*, DNA-TR-95-120, MRC-R-1504, May 1996.
5. Dana, R. A., and R. L. Bogusch, *Digital Communications in Fading Channels: COMLNK Validation*, DSWA-TR-98-61, MRC-R-1531, July 1998.

The first three reports have been published by the Air Force. The fourth report has been published by the Defense Special Weapons Agency (formerly the Defense Nuclear Agency). The fifth report is being published by the Defense Threat Reduction Agency (formerly the Defense Special Weapons Agency). All five reports may be requested through the Defense Technical Information Center.

A.12 YOUR FEEDBACK IS APPRECIATED.

Our goal in developing this software is to provide a tool that facilitates the design and evaluation of digital communications systems that may have to operate in a wide range of channel conditions. We will appreciate your help in evaluating the program in terms of ease of use and suitability to your needs. We want your feedback. Tell us what you like or do not like about the program. Let us know what features you would like to see added. We cannot promise to incorporate every change that may be suggested, but we certainly will consider each suggestion and present a list of potential improvements to our sponsor for his approval.

DISTRIBUTION LIST

DEPARTMENT OF DEFENSE

DEFENSE TECHNICAL INFORMATION CENTER
8725 JOHN J. KINGMAN ROAD, SUITE 0944
FORT BELVOIR, VA 22060-6218
ATTN: DTIC/OCP

DEFENSE THREAT REDUCTION AGENCY
6801 TELEGRAPH ROAD
ALEXANDRIA, VA 22310-3398
ATTN: TDAC, DR L. WITTWER
ATTN: TDANP, DR K. SCHWARTZ
ATTN: TDANP, TRC
ATTN: TDANS

DIRECTOR
DISA
TECHNICAL RESOURCES CENTER
7010 DEFENSE PENTAGON
WASHINGTON, DC 20301-7010
ATTN: JNGO

JOINT CHIEFS OF STAFF
DIRECTOR FOR OPERATIONS, J-3
3000 DEFENSE PENTAGON
WASHINGTON, DC 20318-3000
ATTN: J-36 CCD

US NUCLEAR COMMAND & CONTROL SYS
SUPPORT STAFF
SKYLINE # 3
5201 LEESBURG PIKE, SUITE 500
FALLS CHURCH, VA 22041-3202
ATTN: LIBRARY

DEPARTMENT OF DEFENSE CONTRACTORS

AEROSPACE CORPORATION
P. O. BOX 92957
LOS ANGELES, CA 90009-2957
ATTN: DR J. M. STRAUS, M2/264
ATTN: G. LIGHT, M5/643
ATTN: M. ROLENZ

AUSTIN RESEARCH ASSOCIATES
1101 CAPITAL OF TEXAS HIGHWAY SOUTH
BUILDING B, SUITE 210
AUSTIN, TX 78746
ATTN: M. L. SLOAN
ATTN: R. THOMPSON

INSTITUTE FOR DEFENSE ANALYSIS
CONTROL & DISTRIBUTION
1801 N. BEAUREGARD STREET
ALEXANDRIA, VA 22311
ATTN: H. WOLFHARD

ITT INDUSTRIES
ITT SYSTEMS CORPORATION
ATTN: AODTRA/DTRIAC
1680 TEXAS STREET, SE
KIRTLAND AFB, NM 87117-5669
ATTN: DTRIAC
ATTN: DTRIAC/DARE

JAYCOR
1410 SPRING HILL ROAD, SUITE 300
MCLEAN, VA 22102
ATTN: DR C. P. KNOWLES

LOCKHEED MARTIN CORPORATION
3251 HANOVER STREET
PALO ALTO, CA 94304-1187
ATTN: J. KUMER (UNCLASSIED ONLY)

LOGICON INC.
LOGICON ADVANCED TECHNOLOGY
2100 WASHINGTON BOULEVARD
ARLINGTON, VA 22204-5704
ATTN: D. CARLSON

MISSION RESEARCH CORPORATION
P. O. BOX 1257
HUNTSVILLE, AL 35807
ATTN: B. BAUER

MISSION RESEARCH CORPORATION
P. O. BOX 2256
ATASCADERO, CA 93423-2256
ATTN: R. L. BOGUSCH

MISSION RESEARCH CORPORATION
P. O. BOX 7957
NASHUA, NH 03060
ATTN: R. ARMSTRONG
ATTN: W. WHITE

MISSION RESEARCH CORPORATION
P. O. BOX 542
NEWINGTON, VA 22122-0542
ATTN: B. WORTMAN

MISSION RESEARCH CORPORATION
P. O. DRAWER 719
SANTA BARBARA, CA 93102-0719
ATTN: B. R. MILNER
ATTN: B. ROTH
ATTN: F. GUIGLIANO
ATTN: J. INCERTE
ATTN: R. DANA
ATTN: R. HENDRICK
ATTN: TECH LIBRARY

PACIFIC-SIERRA RESEARCH
OPERATING COMPANY OF VERID
29801 28TH STREET, 2ND FLOOR
SANTA MONICA, CA 90405
ATTN: E. FIELD

PACIFIC-SIERRA RESEARCH CORPORATION
WASHINGTON OPERATIONS
1400 KEY BOULEVARD, SUITE 700
ARLINGTON, VA 22209
ATTN: M. ALLERDING

SCIENCE APPLICATIONS INT'L CORPORATION
2111 EISENHOWER AVENUE, SUITE 205
ALEXANDRIA, VA 22314
ATTN: J. COCKAYNE

SCIENCE APPLICATIONS INTL CORP
P. O. BOX 1303
MCLEAN, VA 22102
ATTN: T. LOVERN

SPARTA INC.
4901 CORPORATE DRIVE, SUITE 102
HUNTSVILLE, AL 35805
ATTN: C. HARPER

TELEDYNE BROWN ENGINEERING
P. O. BOX 070007
HUNTSVILLE, AL 35807-7007
ATTN: J. FORD

TOYON RESEARCH CORPORATION
75 AERO CAMINO, SUITE A
GOLETA, CA 93117-3139
ATTN: J. ISE

TRW INC.
SPACE & ELECTRONICS GROUP
ONE SPACE PARK
REDONDO BEACH, CA 90278-1078
ATTN: T.I.C., S/1930

VISIDYNE, INC.
P. O. BOX 1399
GOLETA, CA 93116-1399
ATTN: J. CARPENTER
ATTN: J. DEVORE
ATTN: J. THOMPSON
ATTN: W. SCHLUETER

DEPARTMENT OF ENERGY

LOS ALAMOS NATIONAL LABORATORY
P. O. BOX 1663
LOS ALAMOS, NM 87545
ATTN: D. WINSKE
ATTN: ESS-5, R. W. WHITAKER, MS-F665

SANDIA NATIONAL LABORATORIES
ATTN: MAIL SERVICES
P.O. BOX 5800
ALBUQUERQUE, NM 87185-0459
ATTN: G. CABLE/MS 0977
ATTN: TECH LIB/MS 0899

DEPARTMENT OF THE AIR FORCE

AIR FORCE FOR STUDIES & ANALYSIS
1570 AIR FORCE PENTAGON
WASHINGTON, DC 20330-1570
ATTN: SATI, ROOM 1D363

AIR FORCE RESEARCH LABATORY
29 RANDOLPH ROAD
HANSCOM AFB, MA 01731-5000
ATTN: VSB/ DR W. BLUMBERG

NATIONAL TEST FACILITY/ENC
ATTN: DATA MANAGEMENT OFFICE
730 IRWEN AVENUE
FALCON AFB, CO 80912-7300
ATTN: NTF/EN, MAJ VAN FOSSON

HEADQUARTERS
US STRATEGIC COMMAND
901 SAC BLVD
OFFUTT AFB, NE 68113-6580
ATTN: DTRA LIAISON
ATTN: J5
ATTN: J61

DEPARTMENT OF THE ARMY

US ARMY NATIONAL GROUND INTELLIGENCE
CENTER
220 7TH STREET, NE
CHARLOTTESVILLE, VA 22901-5396
ATTN: IAFSTC-RMT

DEPARTMENT OF THE NAVY

COMMANDING OFFICER
SPAWARSYSCEN
53560 HULL STREET
SAN DIEGO, CA 92152-5001
ATTN: D822, J FERGUSON

NAVAL RESEARCH LABORATORY
4555 OVERLOOK AVENUE, SW
WASHINGTON, DC 20375-5000
ATTN: CODE 6700, S. OSSAKOW
ATTN: CODE 6790, J. HUBA
ATTN: CODE 7604, H. HECKATHORN

OTHER GOVERNMENT

CENTRAL INTELLIGENCE AGENCY
WASHINGTON, DC 20505
ATTN: ORD/E P G, DR D. CRESS
ATTN: OSWR/SSD FOR, L. BERG

NATIONAL ARCHIVES & RECORDS
ADMINISTRATION
8601 ADELPHI ROAD, ROOM 3360
COLLEGE PARK, MD 20740-6001
ATTN: USER SERVICE BRANCH